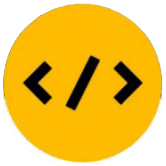
Министерство образования Республики Беларусь
Белорусский государственный университет
Механико-математический факультет
Кафедры веб-технологий и компьютерного моделирования

А. С. Кравчук, А. И. Кравчук, Е. В. Кремень

ООП в языке Java.
Сборник заданий и тематических примеров

Учебные материалы
для студентов специальности: 1-31 03 08
«Математика и информационные технологии
(по направлениям)»



Минск
2023

УДК 004.432.045:004.738.5Java(075.8)
К 772

Решение о депонировании вынес:
Совет механико-математического факультета
Протокол № 6 от 28 февраля 2023 г.

Авторы:
А. С. Кравчук, доктор физико-математических наук профессор кафедры экономической информатики БГЭУ,
А. И. Кравчук, кандидат физико-математических наук доцент кафедры веб-технологий и компьютерного моделирования БГУ,
Е. В. Кремень, кандидат физико-математических наук доцент кафедры веб-технологий и компьютерного моделирования БГУ.

Рецензенты
Кафедра информационных технологий Белорусского государственного экономического университета (заведующая кафедрой Садовская М.Н., кандидат технических наук, доцент);
Медведев С.В., заведующий лабораторией синтеза технических систем Объединенного института проблем информатики НАНБ, доктор технических наук.

Кравчук, А. С. ООП в языке Java. Сборник заданий и тематических примеров : учебные материалы для студентов специальности: 1-31 03 08 «Математика и информационные технологии (по направлениям)» / А. С. Кравчук, А. И. Кравчук, Е. В. Кремень ; БГУ, Механико-математический фак., Каф. веб-технологий и компьютерного моделирования. – Минск : БГУ, 2023. – 150 с. : ил. – Библиогр.: с. 149–150.

Сборник заданий предназначен для проработки приемов объектно-ориентированного программирования в языке Java. Издание содержит задачи на обработку массивов, реализацию интерфейсов, наследование, использование коллекций, дженериков и лямбда-выражений. В каждой теме приводятся примеры решения типовых задач и варианты индивидуальные заданий. Издание ориентировано в первую очередь на тех, кто не имеет опыта практического программирования на языке Java и адресуется студентам, а также всем, кто хотел бы научиться приемам программирования стандартных задач.

ОГЛАВЛЕНИЕ

Введение

В конце прошлого века неуклонное возрастание сложности программных продуктов, при одновременном сохранении требований к их надежности привело к появлению новой парадигмы программирования – объектно-ориентированному программированию.

Эта парадигма в настоящее время является доминирующей при создании как проприетарного, так и свободного программного обеспечения.

Основной концептуальной единицей в ООП любых языков программирования является понятие класса – абстрактного типа, создаваемого самим программистом. Кроме того как следует из названия данной методологии вторым по значимости понятием является объект – собственно переменная (экземпляр), имеющий тип, определяемый уже существующим классом.

В любом языке программирования использующем ООП всегда так же будут использоваться такие принципы, как:

- инкапсуляция – это объединение полей и методов работы с ними в рамках одного абстрактного типа данных (класса), а также сокрытие полей от внешнего доступа;
- наследование – незаменимый и очень гибкий инструмент, применяемый для надежного повторного использовании кода класса предка, с возможностью расширения его функциональности в широком смысле (добавления как новых полей таки методов);
- полиморфизм – концепция позволяющая выполнять одинаковые или близкие по смыслу преобразования с данными разных как базовых, так и абстрактных типов.

Несмотря на то, что эти базовые принципы являются общими для всех языков программирования, использующих ООП, но особенности их реализации существенно зависят от конкретного изучаемого языка программирования. В связи с этим, к сожалению, нет возможности изучить ООП в рамках только одного из языков программировании и в дальнейшем успешно однотипно пользоваться этими значениями в рамках разработки программных продуктов с помощью других языков.

Поэтому выполнение заданий, демонстрирующих специфику применения парадигмы объектно-ориентированного программирования очень важны для изучения каждого языка в отдельности.

Тема 1. Создание класса-библиотеки, содержащего методы обработки одномерных массивов

Тип числовых значений одномерного массива выбирается в соответствии с алгоритмом решаемой задачи.

В некоторых заданиях предполагается, что массивы содержат отрицательные числа. Это достигается выбором границ генерации `MIN` и `MAX`.

1.1 Класс-библиотека, содержащий методы, возвращающие скалярные значения

Обобщенная формулировка задания. Библиотека (класс `Library`) должна содержать пять методов:

- первый – *статический*, проверяющий правильность ввода размерности массива;
- второй – *статический*, инициализирующий массив введенной с клавиатуры размерности случайными целыми числами и возвращающий в качестве значения результат его заполнения;
- третий – *экземплярный*, реализующий задание из ***части 1*** списка индивидуальных заданий и возвращающий значение с помощью ключевого слова `return`;
- четвертый – *экземплярный*, реализующий задание из ***части 2*** списка индивидуальных заданий и возвращающий значение через свойство класса-библиотеки (для метода тип возвращающего значения – `void`);
- пятый – *экземплярный*, геттер для возвращения результата работы четвертого метода.

Пример задания. В библиотеку (класс `Library`) добавить

- *статический* метод, возвращающий значение булевского типа – результат проверки правильности ввода размерности;
- *статический* метод, возвращающий в качестве значения массив заданной размерности, инициализированный случайными целыми числами;
- *экземплярный* метод, определяющий минимальное значение целочисленного массива и возвращающий это значение с помощью ключевого слова `return`;

- *экземплярный* метод, определяющий индекс последнего максимального элемента массива и возвращающий значение через свойство класса-библиотеки (для метода тип возвращающего значения – `void`).
- *экземплярный* метод, геттер для возвращения результата работы четвертого метода.

1.1.1. Пример выполнения задания

Main.java

```
1  import java.util.Arrays;
2  import java.util.Scanner;
3
4  class Library {
5      //свойство для хранения результата работы одного
6      //из методов
7      private int indMax = 0;
8      //проверка ввода размерности
9      static boolean conditionVerify(int n){
10         if (n <= 0) {
11         System.out.println("Число долж. быть больше 0.\n"+
12                                 "Попробуйте еще раз!\n");
13         return false;
14         }
15         return true;
16     }
17     //создание + заполнение массива из случайных чисел
18     static int[] generationArray(final int MIN,
19                                  final int MAX,
20                                  int n) {
21         int[] array = new int[n];
22         for(int i = 0; i < array.length; i++) {
23             array[i] = (int) (Math.random() *
24                     (MAX - MIN) + MIN);
25         }
26         return array;
27     }
28     //экземплярный метод
29     int min(int[] array) {
30         int min = array[0];
31         for(int i = 1; i < array.length; i++) {
32             if ( min > array[i]) min = array[i];
33         }
34         return min;
35     }
36     //экземплярный метод
```

```
37      void maxIndex(int[] array) {
38          for(int i = 1; i < array.length; i++) {
39              if ( array[indMax] <= array[i]) indMax = i;
40          }
41      }
42      //геттер для возвращения результата работы maxIndex()
43      int getMaxInd() {
44          return indMax;
45      }
46  }
47
48  public class Main
49  {
50      public static void main (String args[]) {
51          Scanner in = new Scanner(System.in);
52          int n;
53          // ввод и проверка условий
54          do {
55              System.out.print("Введите натуральное число: ");
56              n = in.nextInt();
57          }
58          while (Library.conditionVerify(n) == false);
59          in.close();
60
61          final int MIN = 0;  // нижняя граница генерации
62          final int MAX = 10; // верхняя граница генерации
63
64          int[] a = Library.generationArray(MIN, MAX, n);
65          System.out.println(Arrays.toString(a));
66          Library obj = new Library();
67          obj.maxIndex(a);
68          System.out.println("Минимальное значение " +
69                                  obj.min(a)+
70                      "\nИндекс последнего максимального "+
71                                  obj.getMaxInd());
72      }
73  }
```

Результат работы программы:

```
Введите натуральное число: 5
[9, 4, 9, 3, 7]
Минимальное значение 3
Индекс последнего максимального 2
```

1.1.2. Список индивидуальных заданий

1.1.2.1. Часть 1

1. Определить сколько раз среди значений одномерного массива встретится максимум.
2. Выяснить сколько из значений массива превышает число z.
3. Найти максимальную сумму двух соседних чисел в массиве.
4. Найти количество значений массива, которые меньше своего левого соседа.
5. Проверить образует ли массив возрастающую или убывающую последовательность.
6. Инициализировать массивы `a`, `b`, `c` из `n` чисел случайными значениями. Понимая, что `a[i]`, `b[i]`, `c[i]` обозначают длины ребер кирпичей, определить кирпич с максимальным объемом.
7. Инициализировать массивы `a`, `b` из `n` чисел случайными значениями. Понимая, что `a[i]`, `b[i]` обозначают длины сторон прямоугольников. Определить прямоугольник с минимальной площадью.
8. Найти среднее арифметическое значение массива и вернуть в качестве возвращаемого значения количество элементов, превышающих среднее арифметическое.
9. Инициализировать массивы `a`, `b` из `n` чисел случайными значениями. Понимая, что `a[i]`, `b[i]` обозначают длины катетов прямоугольных треугольников. Определить треугольник с самой длинной гипотенузой.
10. Инициализировать вектора `a`, `b` из `n` чисел случайными значениями. С помощью метода определить скалярное произведение двух векторов `a`, `b`.
11. Инициализировать массивы `x`, `y` из `n` чисел случайными значениями. Понимая, что `x[i]`, `y[i]` обозначают координаты точек на плоскости, определить среднюю длину радиус-векторов точек.
12. В массиве найти разность между наибольшим и наименьшим значением.
13. Найти среднее арифметическое наибольшего и наименьшего из значений массива.
14. Инициализировать массивы `x`, `y` из `n` чисел случайными значениями. Понимая, что `x[i]`, `y[i]` обозначают координаты точек на плоскости, определить минимальный радиус круга, в который попадают все эти точки.

15. Найти среднее геометрическое значение для массива и вернуть в качестве возвращаемого значения количество элементов, меньших среднего геометрического.

1.1.2.2.Часть 2

Дан одномерный массив `a` из `n` элементов:

1. Определить произведение элементов чье значение без остатка делится на `3` и не делится на `2`.
2. Определить в нем сумму элементов, стоящих на позициях, чей номер больше записанного в них значения.
3. Определить в нем произведение нечетных значений элементов.
4. Определить в нем количество элементов, квадрат которых больше введенного `a`.
5. Определить произведение элементов, которые при делении на `2` дают такой же остаток, как и при делении на `3`.
6. Определить в нем среднее геометрическое четных элементов.
7. Определить в нем количество тех элементов, которые при делении на `3` дают остаток `2`.
8. Определить в нем среднее геометрическое тех элементов, которые при делении на `4` дают остаток `1` или `3`.
9. Определить количество тех элементов, которые без остатка делятся на собственный номер (индекс + `1`).
10. Определить количество тех элементов, стоящих на четных позициях, которые сами четны.
11. Определить среднее арифметическое квадратов элементов, стоящих на позициях, которые при делении на `4` дают остаток `2`.
12. Определить с помощью метода в нем количество элементов кратных четырем.
13. Определить в нем произведение элементов, значения которых лежат вне диапазона `[a; b]`.
14. Определить в нем сумму остатков от деления на `4` тех элементов, которые не кратны `3`.
15. Определить в нем среднее геометрическое элементов, стоящих на четных позициях.
16. Определить среднее арифметическое элементов, стоящих на позициях кратных трем.

1.2. Разработка класса, содержащего методы, возвращающие результат работы по ссылке

Обобщенная формулировка задания. Библиотека (класс `Library`) должна содержать четыре метода:

- первый – *экземплярный*, проверяющий правильность ввода размерности массива;
- второй – *экземплярный*, получающий созданный массив извне и инициализирующий его числами;
- третий – *экземплярный*, реализующий задание из ***части 1*** списка индивидуальных заданий;
- четвертый – *экземплярный*, реализующий задание из ***части 2*** списка индивидуальных заданий;

Пример задания:

- из примера предыдущей темы в библиотеку (класс `Library`) добавить *экземплярный* метод, возвращающий значение булевского типа – результат проверки правильности ввода размерности;
- из примера предыдущей темы в библиотеку (класс `Library`) добавить *экземплярный* метод, инициализирующий массив, созданный вне метода, случайными числами;
- из примера предыдущей темы добавить в библиотеку (класс `Library`) вспомогательный *экземплярный* метод, определяющий минимальное значение в массиве;
- в библиотеку (класс `Library`) добавить *экземплярный* метод делящий весь массив на минимальный, получаемый с помощью вспомогательного.

1.2.1. Пример выполнения задания

Main.java

```
1  import java.util.Arrays;
2  import java.util.Scanner;
3
4  class Library {
5      String report = "Число долж. быть " +
6                      "больше 0.\n" +
7                      "Попробуйте еще раз!\n";
8      //проверка ввода размерности
```

```
    boolean conditionVerify(int n){
        if(n <= 0){
            System.out.println(report);
            return false;
        }
        return true;
    }
    //только заполнение массива случайными числами
    //типа double
    void generationArray(final double MIN,
                         final double MAX,
                         double[] array) {
        for(int i = 0; i < array.length; i++) {
            array[i] = Math.random() *
                       (MAX - MIN) + MIN;
        }
    }
    //вспомагательный экземплярный метод
    double min(double[] array) {
        double min = array[0];
        for(int i = 1; i < array.length; i++) {
            if ( min > array[i]) min = array[i];
        }
        return min;
    }
    //экземплярный метод
    void ArrayDivision(double[] array) {
        Library obj = new Library();
        double min = obj.min(array);
        //исключение деления на нуль
        min = min != 0 ? min : 1;
        for(int i = 0; i < array.length; i++) {
            array[i] = array[i] / min;
        }
    }
}

public class Main
{
    public static void main (String args[]) {
        Scanner in = new Scanner(System.in);
        Library obj = new Library();
        String call = "Введите натуральное число: ";
        int n;
```

```
53
54          // ввод и проверка условий
55          do {
56              System.out.print(call);
57              n = in.nextInt();
58          }
59          while (obj.conditionVerify(n) == false);
60          in.close();
61
62          //создается массив
63          double[] a = new double[n];
64          //границы генерации
65          final double MIN = 0;
66          final double MAX = 10;
67          //заполнение массива
68          obj.generationArray(MIN, MAX, a);
69          System.out.println("Исходный массив\n" +
70                                  Arrays.toString(a));
71          obj.ArrayDivision(a);
72          System.out.println("Результат\n" +
73                                  Arrays.toString(a));
74      }
75  }
```

Результат работы программы:

```
input
Введите натуральное число: 3
Исходный массив
[8.35874839402929, 8.364336715325367, 3.3415593684724 73]
Результат
[2.5014514100494116, 2.503123779347651, 1.0]
```

1.2.2. Список индивидуальных заданий

1.2.2.1. Часть 1

1. Если элемент массива по величине четный, то его нацело разделить на два.
2. Каждый элемент, дающий в остатке двойку при делении на 4 увеличить на единицу.

3. Каждый элемент, чей модуль больше 7 обнулить. Массив вывести до и после преобразования.
4. Каждый четный элемент, в том случае, если он отрицателен, возвести в куб.
5. Каждый элемент с индексом кратным 3, в массиве заменить остатком от деления этого элемента на 5.
6. Каждый элемент, стоящий на четной позиции в массиве заменить остатком от деления этого элемента на 6.
7. Каждый нечетный по величине элемент в массиве заменить остатком от деления этого элемента на 3.
8. Каждый элемент в массиве больший 5 умножить на результат целочисленного деления этого элемента на 5.
9. Каждый элемент в массиве чье значение лежит вне диапазона [−2; 6] увеличить на 12.
10. Каждый элемент массива с индексом кратным 4 умножить на собственный индекс.
11. Каждый элемент в массиве чье значение лежит вне диапазона [−5; 6] возвести в куб.
12. Каждый элемент в массиве чье значение лежит в диапазоне [−1;10] умножить на 3.
13. Каждый второй элемент в массиве, чье значение лежит в диапазоне [−3; 5] заменить 0.
14. Элементы массива, чей квадрат меньше 16 увеличить втрое.
15. К элементам с нечетными значениями массива прибавить значение собственного индекса.
16. Элементы массива, которые при делении нацело на собственный номер (индекс + 1) дают нечетное значение увеличить на 2.

1.2.2.2. Часть 2

1. Найти произведение первых двух положительных элементов массива (предполагается, что они всегда существуют). Произведением заменить все нечетные элементы.
2. Найти наименьший элемент массива среди тех, которые находятся на четных позициях. На полученное значение уменьшить элементы с четными индексами массива.
3. Найти среднее арифметическое Sr максимума и минимума массива. Далее возвести в куб все элементы меньшие чем Sr.
4. Найти произведение элементов массива, принадлежащих интервалу [min/2, max/2]. Значением этого произведения заменить второй и последний элементы массива.

5. Найти индекс первого положительного элемента массва. Все отрицательные элементы, следующие за первым положительным увеличить на модуль суммы отрицательных всего массива.
6. Определить `k` – количество нечетных элементов массива в нем содержащихся. Далее максимальный элемент массива умножить на `k` и снова вывести.
7. Определить максимальный или минимальный элемент в массиве встречается раньше. Если максимальный, то заменить минимумом второй элемент массива, если встречается раньше минимальный, то заменить максимумом предпоследний элемент массива.
8. Определить упорядочены ли элементы массива по возрастанию. Если не упорядочены, то поменять в массиве второй и последний элементы массива, иначе эти элементы возвести в куб.
9. Определить упорядочены ли элементы массива по возрастанию. Если не упорядочены, то определить индекс первого элемента, нарушающего порядок, сам этот элемент увеличить на `2`, иначе поменять местами максимум и минимум в массиве.
10. Найти индекс `FirstEvenIndex` – первого четного значения элемента в массиве. Преобразовать последние `FirstEvenIndex` элементов массива путем их умножения на значение первого нечетного.
11. Вычислить разность между суммой элементов массива, стоящих на четных местах, и суммой элементов, стоящих на нечетных местах. На полученную разность увеличить первую половину массива.
12. Определить количество смен знака (`NumSignChange`) для элементов массива. Если `NumSignChange > 0`, то все элементы после `A[NumSignChange]` заменить значением кубом разностей между первым и минимальным элементами массива.
13. Найти максимум среди элементов первой половины массива и минимум среди второй половины массива, которые поменять местами.
14. Выяснить, какое число в массиве встретится ранее – положительное или отрицательное (нули не рассматривать). Если положительное – найти в массиве максимальный элемент и возвести его в куб, если отрицательное – возвести в квадрат минимальный элемент.
15. Найти минимум среди элементов первой половины массива и максимум среди второй половины. Вычислить сумму найденных значений и заменить им элемент, стоящий перед найденным минимальным элементом (предполагается, что нулевой элемент никогда не будет иметь нулевой индекс).

16. Вычислить наибольшее и наименьшее значения разности между соседними элементами массива. Найденными значениями заменить, соответственно, второй и последний элементы массива.

Тема 2. Основы создания многофайловых проектов. Работа с двумерными массивами

Как и в случае одномерных массивов тип числовых значений выбирается в соответствии с алгоритмом решаемой задачи.

В некоторых заданиях предполагается, что двумерные массивы содержат отрицательные числа. Это достигается выбором границ генерации `MIN` и `MAX`.

2.1. Формирование класса, содержащего методы обработки двумерных массивов хранимых в одномерных

Задание посвящено созданию многофайлового проекта, состоящего из двух файлов (Рисунок 1):

- файла класса `Library`, содержащего методы обработки двумерного массива, «упакованного» в одномерном;
- файла класса `Main`, содержащего метод `main()` (собственно программу, использующую разработанные методы).

Рисунок 1 – Результат создания простейшего многофайлового проекта

Замечание.

В работе используются так называемый безымянный пакет «по умолчанию», т.е. способность интегрированной среды `Online GDB` сформировать многофайловый проект из нескольких файлов без явного создания пакетов и их импорта. Для этого достаточно:

- *файл хранящий `public` класс `Library` назвать `Library.java`;*
- *файл хранящий `public` класс `Main` содержащий программу `main()` оставить названным по умолчанию `Main.java`.*

2.1.1. Создание многофайлового проекта средствами Online GDB

Для создания отдельного файла в Online GDB достаточно нажать на кнопку  New File (Ctrl+M) (Рисунок 2) и в открывшемся окне ввести название RectangularMatrix.java. На этом формирование структуры многофайлового проекта закончено (Рисунок 1).

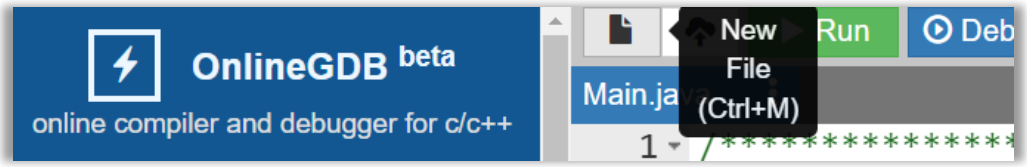


Рисунок 2 – Кнопка добавления файлов

Таким образом остается только заполнить отдельные файлы разработанными публичными классами.

2.1.2. Основы общей теории хранения многомерного массива в одномерном

Работа демонстрирует, что любой ***многомерный*** массив можно ***хранить и обрабатывать как одномерный***. В качестве примера многомерного массива выбран прямоугольный двумерный массив.

Следует обратить ***внимание на организацию индексации при «упаковке»*** двумерного массива в одномерном.

Кроме того, в этом случае придется ***передавать в методы*** в качестве параметров отдельно количество строк и количество столбцов.

Работа с индексами. Пусть, как и раньше индекс i определяет индекс строки, а индекс j – номер столбца.

Будем считать, что все действия с массивом производятся по срокам, т.е. цикл по i является внешним (сама переменная – медленной), тогда цикл по j является внутренним (сама переменная – быстрой). В этом случае результирующий индекс элемента в одномерном массиве, симулирующем двумерный определяется по формуле (Рисунок 3):

$$i \cdot n + j.$$

Сама идея хранения двумерного массива в одномерном состоит в том, что двумерный массив сохраняется построчно в одномерном массиве (Рисунок 3).

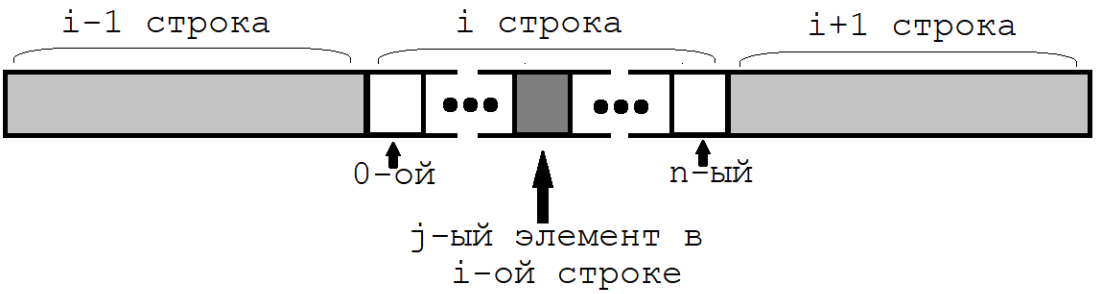


Рисунок 3 – Схема хранения двумерного массива в одномерном

Таким образом обратиться к элементу двумерного массива $m[i, j]$ сохраненному в виде одномерного массива `a` можно с помощью синтаксической конструкции $a[i \cdot n + j]$.

Обобщенная формулировка задания.

Класс `Library` в файле `Library.java` должен содержать пять методов:

- первый – *экземплярный*, проверяющий правильность ввода размерностей двумерного массива (количество строк и количество столбцов);
- второй – *статический*, инициализирующий двумерный массив, «упакованный» в одномерном, случайными целыми числами и возвращающий в качестве значения результат его заполнения;
- третий – *статический*, реализующий задание из ***части 1*** списка индивидуальных заданий;
- четвертый – *статический*, реализующий задание из ***части 2*** списка индивидуальных заданий;
- пятый – *статический*, выводящий двумерный массив, «упакованный» в одномерном.

Пример задания:

- в библиотеку (класс `Library`) добавить *экземплярный* метод, возвращающий значение булевского типа – результат проверки правильности ввода размерностей двумерного массива (количество строк и количество столбцов);
- в библиотеку (класс `Library`) добавить *статический* метод, возвращающий в качестве значения двумерный массив, «упакованный» в одномерном, заданной размерности, инициализированный случайными целыми числами;
- в библиотеку (класс `Library`) добавить *статический* метод, определяющий минимальное значение целочисленного двумерного массива, «упакованного» в одномерном,;

- в библиотеку (класс `Library`) добавить *статический* метод, определяющий величину максимума среди элементов с четными индексами;
- в библиотеку (класс `Library`) добавить *экземплярный* метод, выводящий двумерный массив, «упакованный» в одномерном.

2.1.3. Пример выполнения задания

```
//файл Library.java
```

Main.java | Library.java

```
1  public class Library {
2      public boolean conditionVerify(int n, int m) {
3          String report = "\n n и m долж. быть " +
4                          "больше 0."+
5                          "\n Попробуйте еще раз!\n";
6          if(n <= 0 && m <= 0){
7              System.out.println(report);
8              return false;
9          }
10         return true;
11     }
12     //создание + заполнение одномерного массива,
13     //содержащего матрицу
14     public static int[] generationArray(final int MIN,
15                                         final int MAX,
16                                         int n,
17                                         int m) {
18       int[] array = new int[n * m];
19       for(int i = 0; i < n; i++) {
20          for(int j = 0; j < m; j++) {
21              array[i * m + j] = (int) (Math.random() *
22                                  (MAX - MIN) + MIN);
23          }
24       }
25       return array;
26    }
27     //экземплярный метод, работающий с одномерным
28     //массивом, как с матрицей
29     public static int min(int[] array, int n, int m) {
30         int min = array[0];
31         for(int i = 0; i < n; i++) {
32             for(int j = 0; j < m; j++) {
```

```
                if ( min > array[i * m + j])
                    min = array[i * m + j];
            }
        }
        return min;
    }

    //еще один экземплярный метод, работающий с одномерным
    //массивом как с матрицей
    public static int maxEvenIndex(int[] array,
                                   int n, int m){
        int max = array[0];
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < m; j++) {
                if (i % 2 == 0 &&
                    j % 2 == 0 && max < array[i * m + j]) {
                    max = array[i * m + j];
                }
            }
        }
        return max;
    }

    public static String toString(int[] array,
                                  int n,
                                  int m) {
        String result = "";
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < m; j++) {
                result = result + " " + array[i * m + j];
            }
            result = result + "\n";
        }
        return result;
    }
}
```

//файл Main.java

Main.java | Library.java

```
import java.util.Arrays;
import java.util.Scanner;

public class Main
{
```

```
6       public static void main (String args[]) {
7           Scanner in = new Scanner(System.in);
8           Library obj = new Library();
9           int n, m;
10          // ввод и проверка условий
11          do {
12              System.out.print("Введите натуральные n и m: ");
13              n = in.nextInt();
14              m = in.nextInt();
15          }
16          while (obj.conditionVerify(n, m) == false);
17
18          final int MIN = 0;  // нижняя граница генерации
19          final int MAX = 10; // верхняя граница генерации
20
21          int[] a = Library.generationArray(MIN,MAX,n,m);
22          System.out.println(Library.toString(a, n, m));
23          System.out.println("Минимальное значение " +
24                                  Library.min(a, n, m)+
25              "\nМаксимальное значение на четных индексах "+
26                                  Library.maxEvenIndex(a, n, m));
27      }
28  }
```

Результат работы программы:

```
Введите натуральные n и m: 2 3
 1 5 6
 8 6 8

Минимальное значение 1
Максимальное значение на четных индексах 6
```

2.1.4. Индивидуальные задания

Часть 1

Дан двумерный массив размерностью nxm:

1. Пусть n = m (массив - квадратный). Вычислить наибольшую сумму элементов, стоящих в подстроках ***верхнего треугольника*** двумерного массива (над главной диагональю).

2. Пусть `n` и `m` - четные числа. Вычислить наименьшую сумму модулей элементов «подстолбцов» ***первой четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
3. Определить максимальную сумму только положительных элементов ***столбцов*** двумерного массива.
4. Пусть `n = m` (массив - квадратный). Вычислить наименьшую сумму элементов, стоящих в подстроках ***нижнего треугольника*** двумерного массива (под главной диагональю).
5. Выбрать ***строку*** с минимальной суммой модулей элементов двумерного массива.
6. Пусть `n` и `m` - четные числа. Вычислить наибольшее среднегеометрическое значение элементов подстрок ***второй четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
7. Определить максимальную сумму только положительных элементов ***строк*** двумерного массива.
8. Пусть `n = m` (массив - квадратный). Вычислить наибольшую сумму элементов, стоящих в «подстолбцах» ***верхнего треугольника*** двумерного массива (над главной диагональю)
9. Определить количество ***столбцов*** сумма модулей элементов, которых больших заданного числа `s`.
10. Пусть `n` и `m` - четные числа. Вычислить наибольшую сумму модулей элементов «подстолбцов» ***третьей четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
11. Определить максимальную сумму только элементов среди ***четных*** столбцов двумерного массива.
12. Пусть `n = m` (массив - квадратный). Вычислить наименьшую сумму элементов, стоящих в «подстолбцах» ***нижнего треугольника*** двумерного массива (под главной диагональю).
13. Определить количество ***строк*** сумма модулей элементов, которых больших заданного числа `s`.
14. Пусть `n` и `m` - четные числа. Вычислить наименьшее среднегеометрическое значение элементов ***подстрок четвертой*** четверти двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
15. Выбрать ***столбец*** с минимальной суммой модулей элементов двумерного массива.
16. Пусть `n = m` (массив - квадратный). Вычислить наибольшее скалярное произведение столбца на симметричную относительно ***главной*** диагонали строку.

Часть 2

Дан двумерный массив размерностью `nxm`:

1. Пусть `n = m` (массив - квадратный). Вычислить наименьшее скалярное произведение столбца на симметричную относительно ***побочной*** диагонали строку.
2. Определить максимальное число ***нулевых*** элементов в ***столбцах.***
3. Определить количество локальных ***максимумов среднегеометрических*** значений ***столбцов*** (сумма модулей столбца считается локальным минимумом, когда суммы модулей соседних столбцов больше текущего).
4. Пусть `n` и `m` - четные числа. Определить максимальное число ***отрицательных*** элементов в «подстолбцах» ***третьей четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
5. Определить количество локальных ***минимумов сумм*** модулей ***строк*** (сумма модулей столбца считается локальным минимумом, когда суммы модулей соседних столбцов больше текущего).
6. Определить максимальную сумму только положительных элементов среди ***четных*** строк двумерного массива.
7. Определить максимальное число ***положительных*** элементов в ***столбцах.***
8. Пусть `n` и `m` - четные числа. Определить максимальное число ***нулевых*** элементов в «подстолбцах» ***первой четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
9. Пусть n = m (массив - квадратный). Найти скалярное произведение главной и побочной диагонали
10. Определить минимальное число ***нулевых*** элементов в ***строках.***
11. Определить количество локальных ***максимумов среднегеометрических*** значений ***строк*** (сумма модулей столбца считается локальным минимумом, когда суммы модулей соседних столбцов больше текущего).
12. Пусть `n` и `m` - четные числа. Определить минимальное число ***положительных*** элементов в подстроках ***второй четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).
13. Определить минимальное число ***отрицательных*** элементов в ***строках.***
14. Определить количество локальных ***минимумов сумм*** модулей ***столбцов*** (сумма модулей столбца считается локальным

минимумом, когда суммы модулей соседних столбцов больше текущего).

15. Пусть `n` и `m` - четные числа. Определить минимальное число элементов больше заданного `s` в подстроках ***четвертой четверти*** двумерного массива (четверти двумерного массива нумеруются по часовой стрелке).

2.2. Класс, содержащий методы обработки двумерных массивов, требующие возвращения методом скалярного значения

Задание посвящено созданию многофайлового проекта, состоящего из двух файлов (Рисунок 4):

- файла класса `RectangularMatrix`, содержащего методы обработки двумерного массива, возвращающие скалярное значение;
- файла класса `Main`, содержащего метод `main()` (собственно программу, использующую методы класса `RectangularMatrix`).

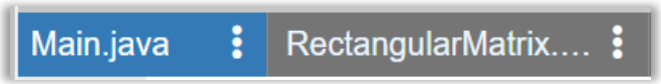


Рисунок 4 – Перечисление файлов текущего проекта

Обобщенная формулировка задания. Библиотека (класс `RectangularMatrix`) должна содержать:

- одно приватное поле для сохранения в ***строковом*** формате результата работы одного из методов из индивидуального задания;
- шесть методов:
  - первый – *статический*, проверяющий правильность ввода размерностей двумерного массива (количество строк и количество столбцов);
  - второй – *статический*, инициализирующий двумерный массив со введенной с клавиатуры размерностью случайными целыми числами и возвращающий в качестве значения результат его заполнения;
  - третий – *экземплярный*, реализующий задание из ***части 1*** списка индивидуальных заданий, возвращающий

значение ***базового*** типа с помощью ключевого слова `return`;

- четвертый – *экземплярный*, реализующий задание из ***части 2*** списка индивидуальных заданий, возвращающий с помощью ключевого слова `return` значение в виде объекта класса `Library` с заполненным в результате работы четвертого метода полем базового типа;
- пятый – *экземплярный* (геттер), позволяющий получить значение поля базового типа объекта класса `Library`;
- шестой – *экземплярный*, преобразующий двумерного массив в строку, гарантирующую ее вывод с помощью метода `print()` или `println()` в виде матрицы.

Пример задания. В библиотеку (класс `RectangularMatrix`) добавить:

- *статический* метод, проверяющий правильность ввода размерности двумерного массива;
- *статический* метод, возвращающий в качестве значения двумерный массив заданной размерности, инициализированный случайными целыми числами;
- *экземплярный* метод, определяющий минимальное значение целочисленного двумерного массива и возвращающий это значение с помощью ключевого слова `return`;
- *экземплярный* метод, определяющий величину максимума среди элементов с четными индексами и возвращающий с помощью ключевого слова `return` значение в виде объекта класса `RectangularMatrix` с заполненным в результате работы этого метода приватным полем `result`;
- *экземплярный* метод (геттер), позволяющий получить значение приватного поля `max` объекта класса `RectangularMatrix`;
- *статический* метод `toString()`, преобразующий двумерный массив в объект типа `String`.

2.2.1. Пример выполнения задания

//файл RectangularMatrix.java

Main.java | RectangularMatrix....

```
1  public class RectangularMatrix {
2      //для хранения результата работы метода
3      //maxOnEvenIndex() в строковом виде;
4      //это гаратирует, что геттер вернет пустую
```

```
    //строку в качестве результата если maxOnEvenIndex()
    //еще не работал
    private String result = "";
    //проверка ввода размерности
    static boolean conditionVerify(int n, int m){
        String report = "\n n и m долж. быть больше 0."+
                                "\n Попробуйте еще раз!\n";
        if(n <= 0 && m <= 0){
            System.out.println(report);
            return false;
        }
        return true;
    }
    //создание + заполнение массива
    static int[][] generationArray(final int MIN,
                                   final int MAX,
                                   int n,
                                   int m) {
        int[][] a = new int[n][m];
        for(int i = 0; i < a.length; i++) {
            for(int j = 0; j < a[i].length; j++) {
                a[i][j] = (int) (Math.random() *
                                    (MAX - MIN) + MIN);
            }
        }
        return a;
    }
    //экземплярный метод, возвращающий значение с
    //помощью return
    int min(int[][] a) {
        int min = a[0][0];
        for(int i = 0; i < a.length; i++) {
            for(int j = 0; j < a[i].length; j++) {
                if ( min > a[i][j]) {
                    min = a[i][j];
                }
            }
        }
        return min;
    }
    //экземплярный метод, возвращающий объект
    //класса Library с заполненным полем max
    RectangularMatrix maxOnEvenIndex(int[][] a) {
        int max = a[0][0];
        for(int i = 0; i < a.length; i++) {
            for(int j = 0; j < a[i].length; j++) {
                if (i % 2 == 0 &&
                    j % 2 == 0 && max < a[i][j]) {
```

```
                    max = a[i][j];
                }
            }
        }
        //приведение результата к строковому типу
        //и присвоение этого значения полю result
        result = new Integer(max).toString();
        //возвращение объекта класса Library
        //с заполненным полем result
        return this;
    }
    //геттер для получения значения поля result
    String getResult() {
        return result;
    }
    //статический метод
    static String toString(int[][] a) {
        String result = "";
        for(int i = 0; i < a.length; i++) {
            for(int j = 0; j < a[i].length; j++) {
                result = result + a[i][j] + " ";
            }
            result = result + "\n";
        }
        return result;
    }
}
```

//файл Main.java

Main.java | RectangularMatrix....

```
import java.util.Scanner;

public class Main
{
    public static void main (String args[]) {
        Scanner in = new Scanner(System.in);
        int n, m;
        // ввод и проверка условий
        do {
            System.out.print("Введите натуральные n и m: ");
            n = in.nextInt();
            m = in.nextInt();
        }
        while (RectangularMatrix.conditionVerify(n, m) == false);
        in.close();
        final int MIN = 0;  // нижняя граница генерации
        final int MAX = 10; // верхняя граница генерации
        int[][] a = RectangularMatrix.generationArray(MIN,MAX,n,m);
        System.out.println(RectangularMatrix.toString(a));
        RectangularMatrix obj = new RectangularMatrix();
```

```
21          System.out.println("Минимальное значение " +
22                                  obj.min(a)+
23              "\nМаксимальное значение на четных индексах "+
24                                  obj.maxOnEvenIndex(a).getResult());
25      }
26  }
```

Результат работы программы:

```
Введите натуральные n и m: 2 3
4 0 2
1 1 0

Минимальное значение 0
Максимальное значение на четных индексах 4
```

2.2.2. Индивидуальные задания

2.2.2.1. Часть 1

Дан двумерный массив размерностью nxm:

1. Определить в нем произведение элементов, квадрат которых меньше 25.
2. Определить в нем сумму модулей положительных элементов.
3. Определить в нем среднее арифметическое модулей отрицательных элементов.
4. Определить в нем среднее геометрическое элементов, у которых оба индекса четные.
5. Определить в нем среднее геометрическое квадратов четных элементов.
6. Определить в нем количество тех элементов, которые при делении на 4 дают остаток 2.
7. Определить в нем произведение тех элементов, чей модуль лежит в диапазоне [15; 50].
8. Определить в нем количество тех элементов, которые без остатка делятся на собственный номер (строки или столбца). Номер – это индекс плюс единица.
9. Определить среднее арифметическое значение из минимального значения с обоими четными индексами и максимального значения с обоими нечетными индексами.

10. Определить в нем среднее арифметическое элементов, значения которых лежат в диапазоне `[2; 20]`.
11. Определить в нем среднее арифметическое элементов, стоящих на позициях с нечетной суммой индексов.
12. Определить в нем сумму элементов чье значение без остатка делится на `2` и не делится на `3`.
13. Определить в нем сумму элементов чье значение без остатка делится на `3` и не делится на `2`.
14. Определить в нем сумму элементов, стоящих на позициях, чьи индексы в сумме больше записанного в них значения.
15. Определить в нем произведение элементов, значения которых лежат вне диапазона `[-10; 25]`.
16. Определить в нем количество элементов, квадрат которых больше `16`.

2.2.2.2. Часть 2

Дан двумерный массив размерностью `nxm`:

1. Определить в нем количество элементов кратных `4`.
2. Определить в нем произведение нечетных элементов.
3. Найти произведение индексов строк и сумму индексов столбцов для элементов массива больших `8`.
4. Определить сумму квадратов индексов (как строк, так и столбцов) элементов, делящихся на `4` без остатка.
5. Определить сумму индексов всех (их может быть несколько) минимальных по величине элементов двумерного массива.
6. Определить в нем количество тех элементов, стоящих на позициях с нечетной суммой индексов, которые сами четны.
7. Определить в нем сумму остатков от деления на `2` тех элементов, которые не кратны `2`.
8. Определить в нем произведение остатков от деления на `4` тех элементов, которые не кратны четырем.
9. Определить в нем среднее геометрическое тех элементов, которые при делении на `4` дают остаток `1` или `3`.
10. Определить в нем произведение тех элементов, чей модуль лежит вне диапазона `[10; 25]`.
11. Найти суммы индексов строк и столбцов нечетных элементов массива.
12. Определить произведение модулей отрицательных элементов.
13. Определить сумму элементов, которые при делении на `2` дают такой же остаток, как и при делении на `3`.

14. Определить сумму элементов, которые при умножении на 3 дают значение большее, чем при возведении в квадрат.
15. Определить среднее геометрическое квадратов элементов, стоящих на позициях, у которых хотя бы один индекс кратен 2.
16. Определить среднее геометрическое квадратов элементов, стоящих на позициях, у которых как минимум один из индексов при делении на 3 дает остаток 2.

2.3. Расширение функциональности проекта. Разработка дополнительного класса, содержащего методы, формирующие квадратный двумерный массив специального вида

Задание посвящено расширению функциональности многофайлового проекта (Рисунок 4), созданного в предыдущем задании. Расширение достигается добавлением дополнительного (третьего) файла `SquareMatrix.java` (Рисунок 5) к исходному проекту (Рисунок 4).

Как и ранее файл `SquareMatrix.java` будет содержать один публичный класс `SquareMatrix`, содержащий методы ***генерации квадратного*** двумерного массива определенного вида.

Рисунок 5 – Перечисление файлов расширенного многофайлового проекта

Обобщенная формулировка задания. Библиотека (класс `SquareMatrix`) должна содержать:

- приватное поле для хранения ссылки на двумерный массив;
- четыре метода:
  - *статический*, проверяющий правильность ввода размерностей квадратного двумерного массива;
  - конструктор `SquareMatrix()`, в котором происходит выделение памяти в куче под хранение квадратного двумерного массива специального вида (напомним, что он будет автоматически проинициализирован нулями);

- *экземплярный* (первый геттер), возвращающий в `main()` ссылку, на массив сформированный согласно заданию из ***части 1***.
- *экземплярный* (второй геттер), возвращающий в `main()` ссылку, на массив сформированный согласно заданию из ***части 2***.

Пример задания. В библиотеку (класс `SquareMatrix`) добавить:

- *статический* метод, возвращающий значение булевского типа – результат проверки правильности ввода размерности ***квадратного*** двумерного массива (количество строк равно количеству столбцов);
- конструктор `SquareMatrix();`
- *экземплярный* метод (первый геттер), размещающий целые числа на главной или побочной диагоналях согласно представленному шаблону:

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & & 1 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 1 & & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

- *экземплярный* метод (второй геттер), заполняющий двумерный массив следующим образом:

$$\begin{pmatrix} 1 & 2 & \cdots & n-1 & n \\ 1 & 2 & & n-1 & n \\ \vdots & & \ddots & \vdots & \\ 1 & 2 & & n-1 & n \\ 1 & 2 & \cdots & n-1 & n \end{pmatrix}$$

2.2.1. Пример выполнения задания

Замечание.

Файл `RectangularMatrix.java` остается без изменений. Поэтому его текст не приводится. Однако класс `Main` существенно изменен и дополнен. Поэтому его новый текст приводится ниже.

```
//RectangularMatrix.java без изменений из предыдущего
//пункта
```

```
//файл SquareMatrix.java
```

Main.java | RectangularMatrix.... | SquareMatrix.java

```java
public class SquareMatrix {
    int a[][] = null;
    //проверка ввода размерности
    static boolean conditionVerify(int n){
        if(n <= 0){
            System.out.println("\n n долж. быть больше 0."+
                               "\n Попробуйте еще раз!\n");
            return false;
        }
        return true;
    }

    SquareMatrix(int n) {
        a = new int[n][n];
    }

    int[][] getLineGenerMatrix() {
        for(int i = 0; i < a.length; i++) {
            a[i][i] = 1;
        }
        return a;
    }

    int[][] getSpecialMatrix() {
        for(int i = 0; i < a.length; i++) {
            for(int j = 0; j < a.length; j++) {
                a[i][j] = j + 1;
            }
        }
        return a;
    }
}
```

```
//файл Main.java
```

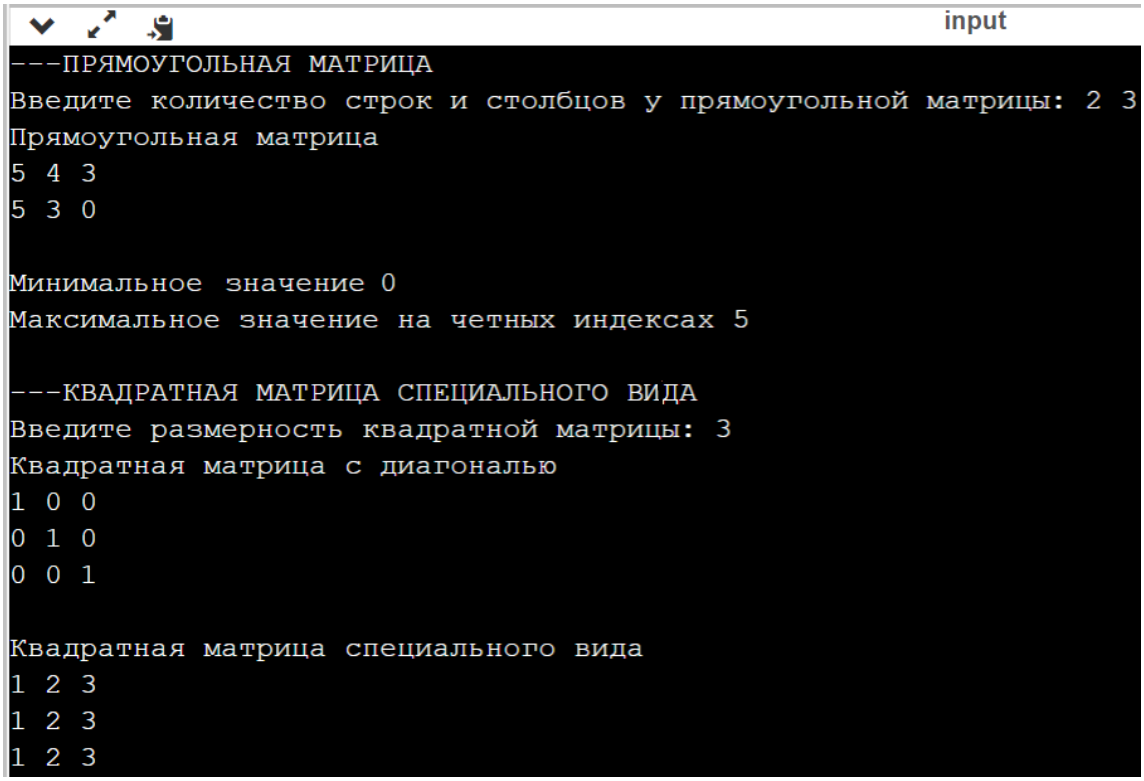
Main.java | RectangularMatrix.... | SquareMatrix.java

```java
import java.util.Scanner;

public class Main
{
    public static void main (String args[]) {
```

```
6           Scanner in = new Scanner(System.in);
7           int n, m;
8           String rectangularChapter =
9                   "---ПРЯМОУГОЛЬНАЯ МАТРИЦА";
10          String rectDimensionsCall = "Введите количество " +
11                                      "строк и столбцов у " +
12                                      "прямоугольной матрицы: ";
13          String squareChapter =
14                  "---КВАДРАТНАЯ МАТРИЦА СПЕЦИАЛЬНОГО ВИДА";
15          String squareDimensionsCall = "Введите размерность " +
16                                        "квадратной матрицы: ";
17          System.out.println(rectangularChapter);
18          //ввод и проверка размерности прямоугольной матрицы
19          do {
20              System.out.print(rectDimensionsCall);
21              n = in.nextInt();
22              m = in.nextInt();
23          }
24          while (RectangularMatrix.conditionVerify(n, m) == false);
25
26          final int MIN = 0;  // нижняя граница генерации
27          final int MAX = 10; // верхняя граница генерации
28
29          int[][] a = RectangularMatrix.generationArray(MIN,MAX,n,m);
30          System.out.println("Прямоугольная матрица");
31          System.out.println(RectangularMatrix.toString(a));
32          RectangularMatrix obj = new RectangularMatrix();
33          System.out.println("Минимальное значение " +
34                             obj.min(a)+
35              "\nМаксимальное значение на четных индексах "+
36                             obj.maxOnEvenIndex(a).getResult());
37          System.out.println("\n" + squareChapter);
38          //ввод и проверка размерности квадратной матрицы
39          do {
40              System.out.print(squareDimensionsCall);
41              n = in.nextInt();
42          }
43          while (SquareMatrix.conditionVerify(n) == false);
44          in.close();
45          System.out.println("Квадратная матрица с диагональю");
46          a = new SquareMatrix(n).getLineGenerMatrix();
47          System.out.println(RectangularMatrix.toString(a));
48          System.out.println("Квадратная матрица специального вида");
49          a = new SquareMatrix(n).getSpecialMatrix();
50          System.out.println(RectangularMatrix.toString(a));
51      }
52  }
```

Результат работы программы:



2.2.2. Индивидуальные задания

Замечание.

Тип числовых значений массива выбирается в соответствии с требуемым шаблоном его формирования.

2.2.2.1. Часть 1

N	Шаблон
1	**2**
1	$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & & 2 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & n-1 & \cdots & 0 & 0 \\ n & 0 & & 0 & 0 \end{pmatrix}$
2	$\begin{pmatrix} n & 0 & \cdots & 0 & 0 \\ 0 & n-1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 2 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$

N	Шаблон
1	**2**
9	$\begin{pmatrix} 0 & 0 & \cdots & 0 & n^2 \\ 0 & 0 & & (n-1)^2 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 2^2 & \cdots & 0 & 0 \\ 1^2 & 0 & & 0 & 0 \end{pmatrix}$
10	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \dots \quad a_{n-1})$ Заполнить по шаблону:

3	$\begin{pmatrix} 0 & 0 & \cdots & 0 & n \\ 0 & 0 & & n-1 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 2 & \cdots & 0 & 0 \\ 1 & 0 & & 0 & 0 \end{pmatrix}$
4	$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & & 2 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 2 & \cdots & 0 & 0 \\ 1 & 0 & & 0 & 0 \end{pmatrix}$
5	$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 2 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & n-1 & 0 \\ 0 & 0 & & 0 & n \end{pmatrix}$

	$\begin{pmatrix} 0 & 0 & \cdots & 0 & a_0 \\ 0 & 0 & & a_1 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & a_{n-2} & \cdots & 0 & 0 \\ a_{n-1} & 0 & & 0 & 0 \end{pmatrix}$
11	$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1^2 \\ 0 & 0 & & 2^2 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 2^2 & \cdots & 0 & 0 \\ 1^2 & 0 & & 0 & 0 \end{pmatrix}$
12	$\begin{pmatrix} n^2 & 0 & \cdots & 0 & 0 \\ 0 & (n-1)^2 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & 2^2 & 0 \\ 0 & 0 & & 0 & 1^2 \end{pmatrix}$
13	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \dots \quad a_{n-1})$ Заполнить по шаблону: $\begin{pmatrix} 0 & 0 & \cdots & 0 & a_{n-1} \\ 0 & 0 & & a_{n-2} & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & a_1 & \cdots & 0 & 0 \\ a_0 & 0 & & 0 & 0 \end{pmatrix}$

1	2
6	$\begin{pmatrix} 1^2 & 0 & \cdots & 0 & 0 \\ 0 & 2^2 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & (n-1)^2 & 0 \\ 0 & 0 & \cdots & 0 & n^2 \end{pmatrix}$
7	Заполнить единицами только **нечетные** строки на побочной диагонали. Шаблон для нечетного n имеет вид: $\begin{pmatrix} 0 & 0 & 0 & & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & & \mathbf{0} & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & \mathbf{0} & & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & \cdots & 0 & 0 & 0 \\ \mathbf{0} & 0 & 0 & & 0 & 0 & 0 \end{pmatrix}$
8	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону: $\begin{pmatrix} a_0 & 0 & \cdots & 0 & 0 \\ 0 & a_1 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & a_{n-2} & 0 \\ 0 & 0 & \cdots & 0 & a_{n-1} \end{pmatrix}$

1	2
14	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону: $\begin{pmatrix} a_{n-1} & 0 & \cdots & 0 & 0 \\ 0 & a_{n-2} & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & a_1 & 0 \\ 0 & 0 & \cdots & 0 & a_0 \end{pmatrix}$
15	Заполнить единицами только **четные** строки на главной диагонали. Шаблон для нечетного n имеет вид: $\begin{pmatrix} \mathbf{1} & 0 & 0 & & 0 & 0 & 0 \\ 0 & \mathbf{0} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{0} & 0 \\ 0 & 0 & 0 & & 0 & 0 & \mathbf{1} \end{pmatrix}$
16	$\begin{pmatrix} 1^2 & 0 & \cdots & 0 & 0 \\ 0 & 2^2 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & 2^2 & 0 \\ 0 & 0 & \cdots & 0 & 1^2 \end{pmatrix}$

2.2.2.2. Часть 2

N	Шаблон
1	2
1	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону (по строкам): $\begin{pmatrix} 0 & 0 & \cdots & 0 & a_0 \\ 0 & 0 & & a_0 & a_1 \\ \vdots & & \ddots & \vdots & \\ 0 & a_0 & & a_{n-3} & a_{n-2} \\ a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \end{pmatrix}$

N	Шаблон
1	2
9	Под главной диагональю: $\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 2 & 1 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ n-2 & n-3 & & 1 & 0 \\ n-1 & n-2 & \cdots & 2 & 1 \end{pmatrix}$

1	2
2	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону (по строкам): $\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ 0 & a_0 & & a_{n-3} & a_{n-2} \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & a_0 & a_1 \\ 0 & 0 & & 0 & a_0 \end{pmatrix}$
3	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону под главной диагональю: $\begin{pmatrix} a_0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ a_{n-2} & a_{n-3} & \cdots & a_0 & 0 \\ a_{n-1} & a_{n-2} & & a_1 & a_0 \end{pmatrix}$
4	Заполнить единицами только **нечетные** ***столбцы*** под побочной диагональю. Шаблон для нечетного n имеет вид: $\begin{pmatrix} 0 & 0 & 0 & & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & \mathbf{0} & & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{pmatrix}$
5	Над главной диагональю: $\begin{pmatrix} 1 & 2 & \cdots & n-1 & n \\ 0 & 1 & & n-2 & n-1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 2 \\ 0 & 0 & & 0 & 1 \end{pmatrix}$

1	2
10	Под главной диагональю: $\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & & 1 & 1 \end{pmatrix}$
11	Над главной диагональю: $\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 0 & 1 & & 1 & 1 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & & 0 & 1 \end{pmatrix}$
12	Над главной диагональю: $\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 0 & 2 & & 2 & 2 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & n-2 & n-2 \\ 0 & 0 & & 0 & n-1 \end{pmatrix}$
13	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону (под побочной диагональю): $\begin{pmatrix} a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \\ a_{n-2} & a_{n-3} & & a_0 & 0 \\ \vdots & & \ddots & \vdots & \\ a_1 & a_0 & \cdots & 0 & 0 \\ a_0 & 0 & & 0 & 0 \end{pmatrix}$

1	2
6	Над побочной диагональю: $\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & & 1 & 0 \\ \vdots & & \ddots & & \vdots \\ 1 & 1 & & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$
7	Над главной диагональю: $\begin{pmatrix} 1 & 2 & \cdots & n-2 & n-1 \\ 0 & 1 & & n-3 & n-2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & 2 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$
8	По строкам: $\begin{pmatrix} 1 & 2 & \cdots & n-2 & n-1 \\ 1 & 2 & & n-2 & n-1 \\ \vdots & & \ddots & & \vdots \\ 1 & 2 & & n-2 & n-1 \\ 1 & 2 & \cdots & n-2 & n-1 \end{pmatrix}$

1	2
14	Заполнить единицами только **четные *строки* и четные *столбцы*** выше главной диагонали. Шаблон для нечетного n имеет вид: $\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & \mathbf{1} & & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & & 0 & 0 & \mathbf{1} \end{pmatrix}$
15	Под главной диагональю: $\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 2 & 1 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ n-2 & n-3 & & 1 & 0 \\ n-1 & n-2 & \cdots & 2 & 1 \end{pmatrix}$
16	Предварительно задан одномерный массив: $(a_0 \quad a_1 \quad \cdots \quad a_{n-1})$ Заполнить по шаблону (под главной диагональю): $\begin{pmatrix} a_0 & 0 & & 0 & 0 \\ a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ a_{n-2} & a_{n-3} & & a_0 & 0 \\ a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \end{pmatrix}$

Тема 3. Реализация интерфейса

3.1. Обработка массивов объектов

Обобщенная формулировка задания. Проект должен состоять из четырех файлов (Рисунок 6):

- первый файл с именем `Main.java` должен содержать программу, работающую с массивом объектов методами класса `Library`;
- второй файл с именем `ИмяКласса.java` должен содержать публичный класс `ИмяКласса`:
  - `ИмяКласса` и имена его полей выбрать в соответствии с индивидуальным заданием;

- класс `ИмяКласса` должен содержать не менее ***семи*** полей;
- в классе `ИмяКласса` в соответствии со структурой его свойств необходимо переопределить метод `toString()` суперкласса `Object`;

- третий файл с именем `Applicable.java` должен содержать публичный интерфейс `Applicable` с тремя абстрактными публичными экземплярными методами:
  - для проверки корректности ввода размерности;
  - для создания массива объектов заданной с клавиатуры размерности;
  - для вывода массива объектов на экран;
- четвертый файл с именем `Library.java` должен содержать публичный класс `Library` реализующий интерфейс `Applicable`, а кроме того, имеющий ***статический*** метод выбора элемента (элементов) из массива по определенному критерию.

Рисунок 6 – Перечисление файлов текущего проекта

Пример задания. На основе предметной области «Человек» (фамилия, имя, отчество, пол, возраст) создать массив объектов. Вывести на экран содержимое массива. Сделать выборку в исходном массиве согласно полу и возрастному диапазону.

3.1.1. Пример выполнения задания

```
//Файл Person.java
```

Main.java | Person.java | Applicable.java | Library.java

```
1  public class Person {
2      //свойства
3      private String firstName, secondName, lastName;
4      private int age;
5      private char gender;
6      //методы
7      public Person(String firstName, String secondName,
8              String lastName, int age, char gender) {
9          this.firstName = firstName;
10         this.secondName = secondName;
```

```
        this.lastName = lastName;
        this.age = age;
        this.gender = gender;
    }
    public String getFirst()  {return firstName;}
    public String getSecond() {return secondName;}
    public String getLast()   {return lastName;}
    public int getAge()       {return age;}
    public char getGender()   {return gender;}
  @Override
  public String toString() {
    return "ФИО " + this.getLast() +
           " " + this.getFirst().charAt(0)+ "." +
           this.getSecond().charAt(0)+ "." +
           "   возраст: " + this.getAge() + ", пол: " +
           this.getGender();
  }
}
```

//Файл Applicable.java

Main.java | Person.java | Applicable.java | Library.java

```
public interface Applicable {
   abstract public boolean conditionVerify(int n);
   abstract public Person[] generationArray(int n);
   public void display(Person[] array);
}
```

//Файл Library.java

Main.java | Person.java | Applicable.java | Library.java

```
import java.util.Scanner;
public class Library implements Applicable {
   @Override
   public boolean conditionVerify(int n) {
      if(n <= 0){
        System.out.println("\n n долж. быть больше 0."+
                           "\n Попробуйте еще раз!\n");
        return false;
      }
      return true;
   }
   @Override
   public Person[] generationArray(int n) {
```

```
14          Scanner in = new Scanner(System.in);
15          Person[] array = new Person[n];
16          for(int i = 0; i < n; i++) {
17              System.out.println("Ввод " + (i + 1) +
18                                 "-ой записи:");
19              System.out.println("\tФамилия: ");
20              System.out.print("\t  ");
21              String lastName = in.nextLine();
22              System.out.println("\tИмя: ");
23              System.out.print("\t  ");
24              String firstName = in.nextLine();
25              System.out.println("\tОтчевство: ");
26              System.out.print("\t  ");
27              String secondName = in.nextLine();
28              System.out.println("\tВозраст: ");
29              System.out.print("\t  ");
30              int age = Integer.parseInt(in.nextLine());
31              System.out.println("\tПол: ");
32              System.out.print("\t  ");
33              char gender = in.nextLine().charAt(0);
34              array[i] = new Person(firstName,
35                                    secondName,
36                                    lastName, age, gender);
37          }
38          return array;
39      }
40      @Override
41      public void display(Person[] array){
42          for(int i = 0; i < array.length; i++) {
43                  System.out.println(array[i].toString());
44          }
45      }
46      //статический метод, работающий с массивом объектов
47      //Person
48      public static void resultChoise(Person[] array,
49                                      int minAge, int maxAge){
50          int age;
51          for(int i = 0; i < array.length; i++) {
52              age = array[i].getAge();
53              if (minAge < age && age < maxAge)
54                  System.out.println(array[i].toString());
55          }
56      }
57  }
```

```
//Файл Main.java
```

Main.java | Person.java | Applicable.java | Library.java

```
1  import java.util.Scanner;
2
3  public class Main
4  {
5      public static void main (String args[]) {
6          Library obj = new Library();
7          Scanner in = new Scanner(System.in);
8          int n;
9          // ввод и проверка условий
10         do {
11             System.out.print("Введите натуральное n: ");
12             n = Integer.parseInt(in.nextLine());
13         }
14         while (obj.conditionVerify(n) == false);
15
16         Person[] a = obj.generationArray(n);
17         System.out.println("Исходная база данных:");
18
19         obj.display(a);
20         System.out.println("Введ. нижнюю границу возраста:");
21         int minAge = Integer.parseInt(in.nextLine());
22         System.out.println("Введ. верхнюю границу возраста:");
23         int maxAge = Integer.parseInt(in.nextLine());
24         in.close();
25         System.out.println("Результат выбора");
26         Library. resultChoise(a, minAge, maxAge);
27     }
28 }
```

Результат работы программы

```
Введите натуральное n: 2
Ввод 1-ой записи:
        Фамилия:
          Иванов
        Имя:
          Иван
        Отчевство:
          Иванович
```

```
        Возраст:
          49
        Пол:
          м
Ввод 2-ой записи:
        Фамилия:
          Сидорова
        Имя:
          Акулина
        Отчевство:
          Федотовна
        Возраст:
          37
        Пол:
          ж
Исходная база данных:
ФИО Иванов И.И.   возраст: 49, пол: м
ФИО Сидорова А.Ф.   возраст: 37, пол: ж
Введ. нижнюю границу возраста:
30
Введ. верхнюю границу возраста:
40
Результат выбора
ФИО Сидорова А.Ф.   возраст: 37, пол: ж
```

3.1.2. Варианты индивидуальных заданий

1. Класс «Школьник» и его возможные свойства: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс.
2. Класс «Студент» и его возможные свойства: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); ВУЗ; курс; группа; средний бал; специальность.
3. Класс «Покупатель» и его возможные свойства: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц

число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета.

4. Класс «Пациент» и его возможные свойства: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови.
5. Класс «Владелец автомобиля» и его возможные свойства: фамилия; имя; отчество; номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира) марка автомобиля; номер автомобиля; номер техпаспорта.
6. Класс «Военнослужащий» и его возможные свойства: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); должность; звание.
7. Класс «Рабочий» и его возможные свойства: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); № цеха; табельный номер; образование; год поступления на работу.
8. Класс «Владелец телефона» и его возможные свойства: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона.
9. Класс «Абитуриент» и его возможные свойства: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); оценки по экзаменам; проходной балл.
10. Класс «Государство» и его возможные свойства: название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства.
11. Класс «Автомобиль» и его возможные свойства: марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.
12. Класс «Товар» и его возможные свойства: наименование; стоимость; срок хранения; сорт; дата выпуска; срок годности; артикул.
13. Класс «Кинолента» и его возможные свойства: название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль.
14. Класс «Рейс» и его возможные свойства: марка автомобиля; номер автомобиля; пункт назначения; грузоподъемность (в тоннах); стоимость единицы груза; общая стоимость груза.

15. Класс «Книга» и его возможные свойства: название; автор (фамилия; имя); год выхода; издательство; себестоимость; цена; прибыль.
16. Класс «Здание» и его возможные свойства: адрес; тип здания; количество этажей; количество квартир; срок эксплуатации; срок до капитального ремонта (25 лет - срок эксплуатации).

3.2. Создание односвязного списка объектов

Обобщенная формулировка задания. Проект должен состоять из ***шести*** файлов (Рисунок 7):

- первый файл с именем `Main.java` должен содержать программу, работающую со списком объектов методами класса `UserList`;
- второй файл с именем `ИмяКласса.java` должен содержать публичный класс `ИмяКласса`:
  - `ИмяКласса` и имена его полей выбрать в соответствии с индивидуальным заданием;
  - класс `ИмяКласса` должен содержать не менее *семи* полей;
  - в классе `ИмяКласса` в соответствии со структурой его свойств необходимо переопределить метод `toString()` суперкласса `Object`;
- третий файл с именем `NodeИмяКласса.java` должен содержать публичный класс `NodeИмяКласса` добавляющий к объекту класса `ИмяКласса` поле `next`, сеттеры и геттеры, необходимые для вязки односвязного списка, а также переопределенный метод `toString()`;
- четвертый файл с именем `Applicable.java` должен содержать публичный интерфейс `Applicable` с одним абстрактным публичным экземплярным методом для выборки элементов из списка по признаку;
- пятый файл с именем `UserList.java`, должен содержать класс реализующий интерфейс `Applicable`, а кроме того, иметь:
  - приватное поле `begin`;
  - четыре метода:
    - конструктор инициализирующий список из `n` элементов;

- метод, инициализирующий вводимыми с клавиатуры значениями поля исходного класса `ИмяКласса`;
- метод, выбирающий записи из списка по признаку;
- переопределенный метод `toString()`.

- шестой файл `Library.java`, содержащий статический метод проверки корректности введенного числа элементов в списке.

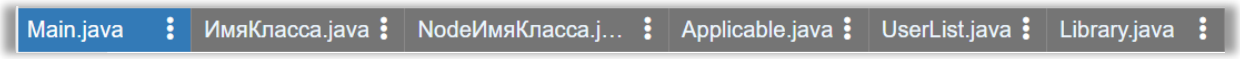


Рисунок 7 – Перечисление файлов текущего проекта

Пример задания. На основе предметной области «Человек» (фамилия, имя, отчество, пол, возраст) создать односвязный список объектов. Вывести на экран содержимое списка. Сделать выборку в исходном списке согласно полу.

3.2.1. Пример выполнения задания

```
//Файл Person.java
```

Main.java | Person.java | NodePerson.java | Applicable.java | UserList.java | Li

```
1  public class Person {
2      //свойства
3      private String firstName, secondName, lastName;
4      private int age;
5      private char gender;
6      //методы
7      public Person(String firstName, String secondName,
8              String lastName, int age, char gender) {
9          this.firstName = firstName;
10         this.secondName = secondName;
11         this.lastName = lastName;
12         this.age = age;
13         this.gender = gender;
14     }
15     public String getFirst()  {return firstName;}
16     public String getSecond() {return secondName;}
17     public String getLast()   {return lastName;}
18     public int getAge()       {return age;}
19     public char getGender()   {return gender;}
20     @Override
21     public String toString() {
22         return "ФИО " + this.getLast() +
```

```
                " " + this.getFirst().charAt(0)+ "." +
                 this.getSecond().charAt(0)+ "." +
                 "   возраст: " + this.getAge() + ", пол: " +
                 this.getGender();
    }
}
```

//Файл Node.java

Main.java | Person.java | NodePerson.java | Applicable.jav

```
public class NodePerson {
    private Person obj;
    private NodePerson next = null;
    //Методы
    NodePerson(Person obj) {
        this.obj = obj;
    }
    public NodePerson getNext() {
        return next;
    }
    public void setNext(NodePerson obj) {
        this.next = obj;
    }
    public Person getPerson() {
        return obj;
    }
    @Override
    public String toString() {
        return obj.toString();
    }
}
```

//Файл Applicable.java

Main.java | Person.java | NodePerson.java | Applicable.java | UserList.ja

```
public interface Applicable {
    abstract public String resultChoise(char gender);
}
```

//Файл UserList.java

Main.java | Person.java | NodePerson.java | Applicable.java | UserList.java | Librar

```
import java.util.Scanner;

public class UserList implements Applicable {
    //адрес начала односвязного списка
```

```
5       private NodePerson begin;
6       public NodePerson getBegin() {return begin;}
7       //конструктор инициализирующий список из n
8       //записей
9       public UserList(int n) {
10          System.out.println("Ввод 1-ой записи:");
11          begin = new NodePerson(initNode());
12          NodePerson current = begin;
13          for(int i = 1; i < n; i++) {
14              System.out.println("Ввод " + (i + 1) +
15                                 "-ой записи:");
16              current.setNext(new NodePerson(initNode()));
17              current = current.getNext();
18          }
19      }
20      //приватный экземплярный метод заполнения одного
21      //узла списка
22      private Person initNode() {
23          Scanner in = new Scanner(System.in);
24          System.out.println("\tФамилия: ");
25          System.out.print("\t  ");
26          String lastName = in.nextLine();
27          System.out.println("\tИмя: ");
28          System.out.print("\t  ");
29          String firstName = in.nextLine();
30          System.out.println("\tОтчевство: ");
31          System.out.print("\t  ");
32          String secondName = in.nextLine();
33          System.out.println("\tВозраст: ");
34          System.out.print("\t  ");
35          int age = Integer.parseInt(in.nextLine());
36          System.out.println("\tПол: ");
37          System.out.print("\t  ");
38          char gender = in.nextLine().charAt(0);
39          return new Person(firstName,
40                            secondName,
41                            lastName, age, gender);
42      }
43      //экземплярный метод, выбора по критерию
44      public String resultChoise(char gender) {
45          String result = "";
46          NodePerson current = begin;
47          while (current != null) {
48              if (current.getPerson().getGender() == gender) {
49                  result = result + current.toString() + "\n";
50              }
51              current = current.getNext();
52          }
```

```
        return result;
    }
    //метод вывода списка на экран
    @Override
    public String toString() {
        String result = "";
        NodePerson current = this.begin;
        while (current != null) {
            result = result + current.toString() + "\n";
            current = current.getNext();
        }
        return result;
    }
}
```

//Файл с классом Library.java

Person.java | NodePerson.java | Applicable.java | UserList.java | Library.java

```
public class Library {
    //метод проверяющий корректность ввода количества
    //узлов в списке
    public static boolean conditionVerify(int n) {
        if(n <= 0){
            System.out.println("\n n долж. быть больше 0."+
                               "\n Попробуйте еще раз!\n");
            return false;
        }
        return true;
    }
}
```

//Файл с классом Main.java содержащем программу

Main.java | Person.java | NodePerson.java | Applicable.java | UserList.java

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        // ввод и проверка условий
        do {
            System.out.print("Введите натуральное n: ");
            n = Integer.parseInt(in.nextLine());
        }
        while (Library.conditionVerify(n) == false);
```

```
14          //создание списка
15          UserList list = new UserList(n);
16          System.out.println("Исходный список:");
17          System.out.println(list.toString());
18          System.out.print("Введите пол для выборки: ");
19          char gender = in.nextLine().charAt(0);
20          System.out.println("Результат выбора");
21          System.out.println(list.resultChoise(gender));
22      }
23  }
```

Результат работы программы:

```
Введите натуральное n: 2
Ввод 1-ой записи:
        Фамилия:
          Иванов
        Имя:
          Иван
        Отчевство:
          Иванович
        Возраст:
          29
        Пол:
          м
Ввод 2-ой записи:
        Фамилия:
          Петрова
        Имя:
          Зинаида
        Отчевство:
          Михайловна
        Возраст:
          43
        Пол:
          ж
Исходный список:
ФИО Иванов И.И.   возраст: 29, пол: м
ФИО Перотрова З.М.   возраст: 43, пол: ж
```

```
Введите пол для выборки: ж
Результат выбора
ФИО Перотрова З.М.   возраст: 43, пол: ж
```

3.2.2. Индивидуальные задания

1. Класс «Учебная дисциплина» и его возможные свойства: название; продолжительность в часах лекционного курса; продолжительность в часах практических занятий; продолжительность в часах лабораторных занятий; количество тематических разделов, тип (общеобразовательная или дисциплина специализации).
2. Класс «Автобус» и его возможные свойства: фирма производитель, классификация по назначению, классификация по длине, классификация по компоновке, классификация по типу двигателя, параметры булевского типа, указывающие наличие: пневмоподвески, автоматических дверей, пандуса для инвалидов, ремней безопасности для крепления грузов.
3. Класс «Трамвай» и его возможные свойства: фирма производитель, разновидность по решаемым задачам, скорость передвижения, компоновка трамвая, схема управления двигателем, вместимость (количество пассажиров), требуемая инфраструктура.
4. Класс «Троллейбус» и его возможные свойства: фирма производитель, разновидность по решаемым задачам, компоновка шасси, тип тягового электродвигателя, система управления двигателем, наличие системы автономного хода, тип тормозной системы, тип вентиляции салона.
5. Класс «Мебель» и его возможные свойства: название, комплектность, фирма-производитель, год изготовления, назначение, материал изготовления, способ изготовления, стиль.
6. Класс «Подвижной состав автомобильного транспорта» и его возможные свойства: тип транспортного средства, фирма изготовитель, год выпуска, дата последнего технического осмотра, тип двигателя, назначение, проходимость, приспособленность к климатическим условиям, характер использования.
7. Класс «Проездной билет» и его возможные свойства: город, тип (разовый или многоразовый (абонементный)), носитель (БСК или бумажный), ограничение времени действия, вид транспорта, ограничение валидности по количеству поездок.
8. Класс «Рейс» и его возможные свойства: тип транспорта, класс обслуживания, пункт отправления, время отправления, пункт назначения, время прибытия, время в пути, количество остановок.

9. Класс «Дерево» и его возможные свойства: вид листьев, срок жизни листьев, род дерева, распространение, форма кроны, наличие плодов, характеристика корневой системы.
10. Класс «Медицинские расходные материалы» и его возможные свойства: вид, назначение, указание о стерильности, материал изготовления, фирма изготовитель, дата изготовления, срок годности.
11. Класс «Фрукт» и его возможные свойства: вид, вкус, размер, цвет кожуры, цвет мякоти, региональная принадлежность, сезонность.
12. Класс «Хлебобулочное изделие» и его возможные свойства, классифицирующие объект по: виду муки, рецептуре, способу выпечки, типу теста, по массе, группе изделия.
13. Класс «Электростанция» и его возможные свойства, классифицирующие объект в зависимости от: источника энергии (в частности, вида топлива), типа силовой установки, мобильности, степени применения (распространенности), мощности генерации, срока службы, ожидаемой стоимости сооружения.
14. Класс «Программист» и его возможные свойства: ФИО, ВУЗ по диплому, специальность по диплому, языки программирования, опыт работы, профессиональный уровень, перечисление проектов в работе над которыми принимал участие.
15. Класс «Животное» и его возможные свойства: название, срок жизни, тип, класс, отряд, семейство, род, вид.
16. Класс «Принтер» и его возможные свойства и его возможные свойства, классифицирующие объект по: названию, фирме-производителю, стране происхождения, возможности печати графической информации, конструктивному устройству и принципу формирования изображения, количеству выдаваемых цветов, типу интерфейса подключения.

Тема 4. Наследование

4.1. Применение наследования для расширения функциональности односвязного списка

4.1.1. Разработка класса, содержащего методы добавления/удаления новых узлов списка

Обобщенная формулировка задания. Предполагается, что в соответствии с заданием из пункта «3.2. Создание односвязного списка объектов» проект ***уже имеет шесть*** файлов (Рисунок 7):

1. `Main.java`;
2. `ИмяКласса.java`;
3. `NodeИмяКласса.java`;
4. `Applicable.java`;
5. `UserList.java`,
6. `Library.java`.

Замечание.

В уже созданном проекте из параграфа «Создание односвязного списка объектов» в файле `UserList.java` *должны быть заменены спецификаторы доступа* `private` *на* `protected`*.*

В соответствии с текущим заданием упомянутый проект из предыдущего задания (Рисунок 7) дополняется ***седьмым файлом*** `AdvancedList.java`, содержащем публичный класс `AdvancedList`, ***расширяющий*** класс `UserList` и состоящий из ***экземплярных*** методов редактирования односвязного списка (Рисунок 8):

- добавления элемента в начало списка;
- добавления элемента в конец списка;
- ***добавления*** элемента в середину списка ***по признаку*** (признак из части 1 раздела «Индивидуальные задания» данной темы);
- удаления элемента списка из его начала;
- удаления последнего элемента списка;
- ***удаления*** элемента списка ***по признаку*** из середины по признаку (признак из части 2 раздела «Индивидуальные задания» данной темы).

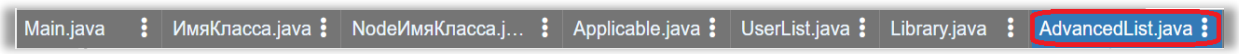


Рисунок 8 – Перечисление файлов текущего проекта

Условия демонстрации работоспособности задания. При демонстрации работоспособности задания при исходном формировании списка он должен быть проинициализирован не менее чем ***тремя*** записями (нодами).

После этого необходимо вызвать в `main()` последовательно все методы класса `AdvancedList`.

4.1.1.1. Теоретические основы редактирования односвязного списка.

4.1.1.1.1. Добавление элемента списка в начало

Следует помнить, что при вставке элемента в начало (первую позицию) списка необходимо также изменить адрес входа в список. Графически вставка нового элемента списка выглядит очевидно (Рисунок 9):

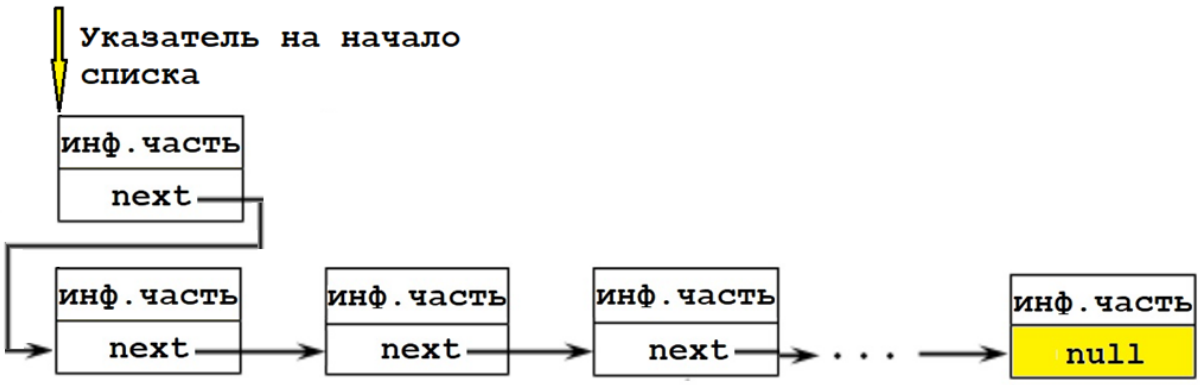


Рисунок 9 – Графическая схема вставки элемента односвязного списка в его начало

```
public static Node addNodeBegin(Node beginList) {
    //создание и заполнение полей нового элемента
    Node current = Library.initNode();
    //связывание нового первого элемента с
    //остальным списком и изменение адреса начала
    //дополненного списка
    current.setNext(beginList);
    return current;
}
```

4.1.1.1.2. Добавление элемента списка в конец

Добавление элемента списка в его конец не представляет на самом деле никаких сложностей. Это один из наиболее очевидных алгоритмов. Единственной особенностью является то, что разработчик должен не забыть, что поле `next` вновь добавленного элемента должно иметь значение `null`, что является признаком окончания списка. Графическая интерпретация добавления элемента в конец списка приведена на рисунке

(Рисунок 10). Прежде чем добавить элемент в конец списка необходимо его весь пройти от начала до последнего элемента, к которому будет присоединяться новый.

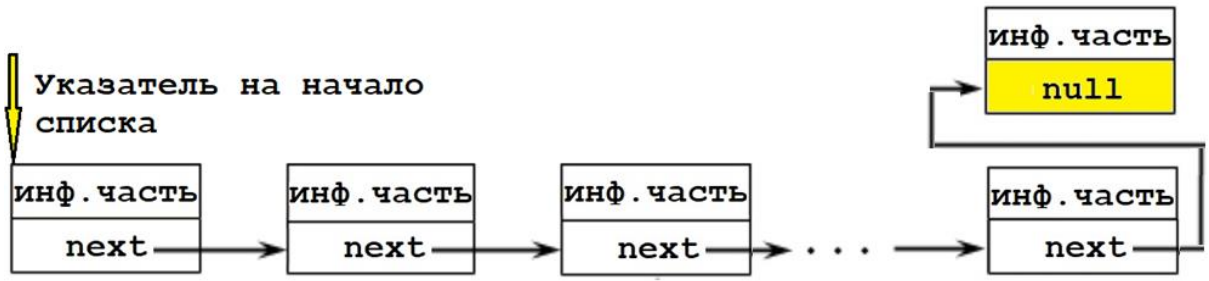


Рисунок 10 – Графическая схема вставки элемента односвязного списка в его конец

```
public static void addNodeEnd(Node beginList) {
    Node current = beginList;
    //движение по списку до его конца
    while (current.getNext() != null) {
       current = current.getNext();
    }
    //создание и заполнение нового последнего
    //элемента, а также присоединение его к
    //списку
    current.setNext(Library.initNode());
}
```

4.1.1.1.3. Добавление элемента списка в середину (не крайние элементы) по признаку

Особняком стоит алгоритм добавления элемента списка в его середину. В отличие от предыдущих случаев, очевидно, у вставляемого элемента будет как предыдущий элемент, так и следующий (Рисунок 11).

В примере рассматривается случай вставки нового элемента односвязного списка после элемента списка, имеющего заданный номер, передаваемый в качестве параметра в метод.

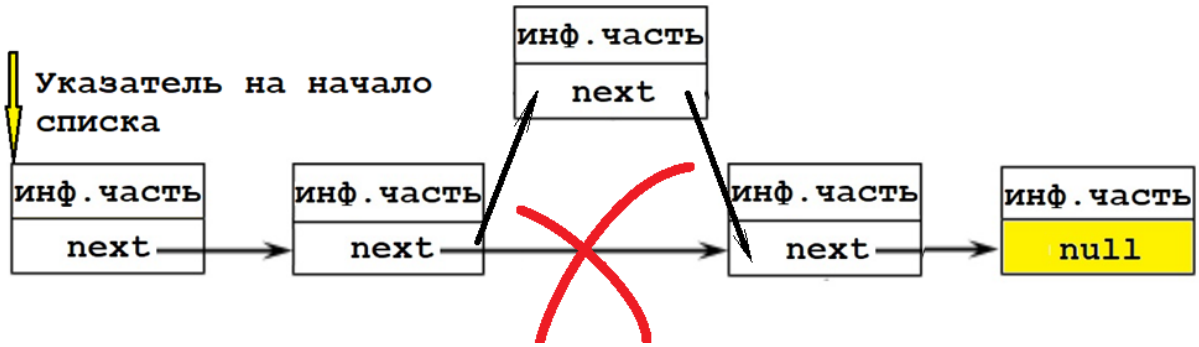


Рисунок 11 – Графическая схема вставки нового срединного элемента в односвязный список

```
public static void addNodeTag(Node beginList, int number){
    Node current = beginList;
    for(int i = 1; i < number - 1; i++) {
        current = current.getNext();
    }
    Node nextElement = current.getNext();
    //создание нового среднего элемента и
    //вставка его в список
    current.setNext(Library.initNode());
    current.getNext().setNext(nextElement);
}
```

4.1.1.1.4. Удаление элемента списка из его начала

Удаление элемента односвязного списка из его начала имеет, очевидную особенность. Необходимо изменить адрес начала списка на один элемент. Графическая схема данной операции приведена на рисунке (Рисунок 12).

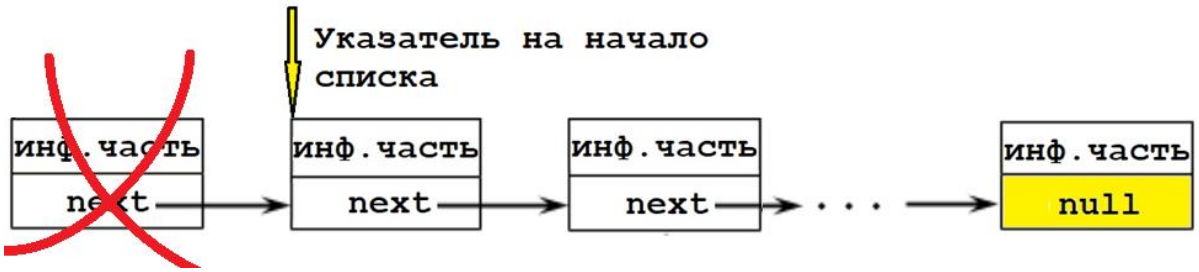


Рисунок 12 – Удаление элемента односвязного списка из его начала

```
public static Node removeNodeBegin(Node beginList) {
    return beginList.getNext();
}
```

4.1.1.1.5. Удаление последнего элемента односвязного списка

Также как и в случае с добавлением элемента в конец списка единственной особенностью данного алгоритма является то, что необходимо пройти весть список до конца, удалить последний элемент, а полю `next` предпоследнего элемента следует присвоить адрес `null` (Рисунок 13).

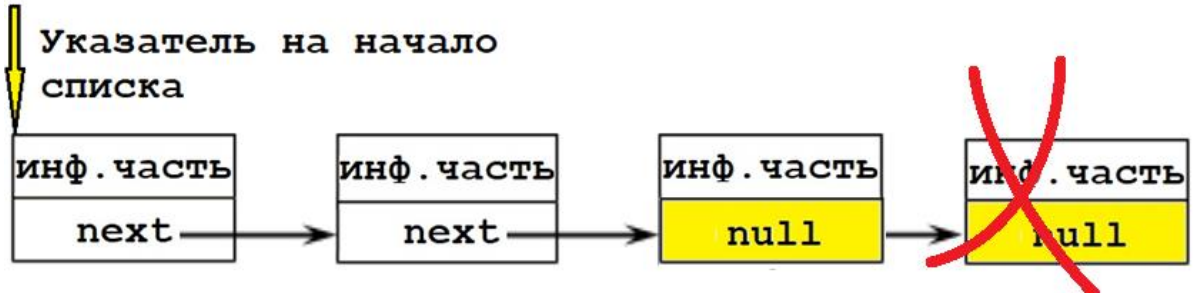


Рисунок 13 – Удаление последнего элемента списка

```
public static void removeNodeEnd(Node beginList) {
    Node current = beginList;
    //перемещение по списку до предпоследнего эл.
    while (current.getNext().getNext() != null) {
        current = current.getNext();
    }
    //присвоение значения null полю next
    //предпоследнего элемента
    current.setNext(null);
}
```

4.1.1.1.6. Удаление элемента списка из середины (не крайних элементов) по признаку

Удаление элемента списка связано с изменением адресов поля next у предыдущего. Вместо адреса текущего элемента это поле должно получить значение следующего (Рисунок 14). Далее будет рассматриваться пример удаление элемента списка по его номеру (не индексу).

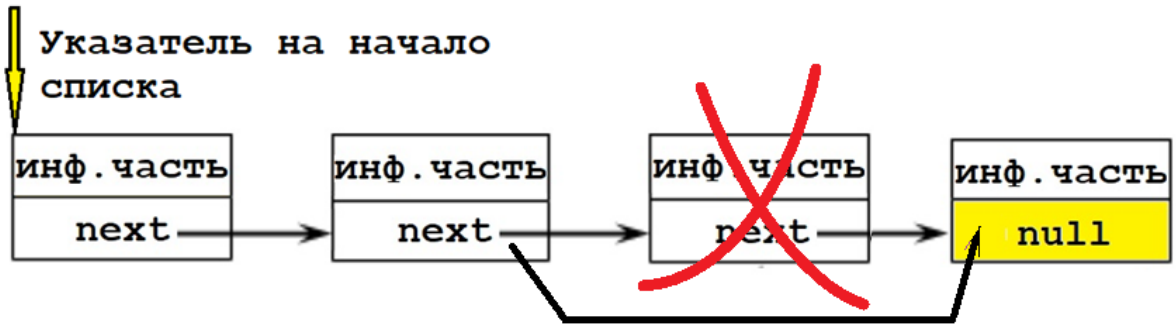


Рисунок 14 – Удаление элемента списка из его середины

```
public static void removeNodeTag(Node beginList, int number) {
    Node current = beginList;
    for(int i = 1; i < number - 1; i++) {
        current = current.getNext();
    }
    current.setNext( current.getNext().getNext() );
}
```

4.1.1.2. Пример выполнения задания

```
//шесть файлов без изменений из проекта раздела
//«Создание односвязного списка объектов»

//Файл AdvancedList.java
```

odePerson.java | Applicable.java | UserList.java | Library.java | AdvancedList.java

```
1  public class AdvancedList extends UserList {
2      //конструктор
3      AdvancedList(int n) {
4          super(n);
5      }
6      //добавление элемента в начало
7      void addNodeBegin() {
8          //создание и заполнение полей нового элемента
9          NodePerson current = new NodePerson(initNode());
10         //связывание элементов
11         current.setNext(begin);
12         //новое в начало списка
13         begin = current;
14     }
15     //добавление элемента в конец
16     void addNodeEnd() {
17         if (begin == null){
18             //если списка не существует
19             begin = new NodePerson(initNode());
```

```
20              return;
21          }
22          //если список уже создан
23          NodePerson current = begin;
24          //движение по списку до его конца
25          while (current.getNext() != null) {
26              current = current.getNext();
27          }
28          //создание и заполнение нового последнего
29          //элемента, а также присоединение его к
30          //списку
31          current.setNext(new NodePerson(initNode()));
32      }
33      //добавление элемента в середину
34      void addNodeTag(int n) {
35          if (begin == null){
36              //если списка не существует
37              begin = new NodePerson(initNode());
38              return;
39          }
40          //движение по списку до конца или до ук. номера
41          NodePerson current = begin;
42          for(int i = 2;
43              current.getNext() != null && i < n;
44              i++) {
45              current = current.getNext();
46          }
47          //создание нового элемента и его вставка в список
48          NodePerson newNode = new NodePerson(initNode());
49          newNode.setNext(current.getNext());
50          current.setNext(newNode);
51      }
52      //удаление элемента из начала
53      void removeNodeBegin() {
54          if (begin != null) {
55              begin = begin.getNext();
56              return;
57          }
58          return;
59      }
60      //удаление последнего элемента списка
61      void removeNodeEnd() {
62          //элементов в списке нет
63          if (begin == null) return;
64          //в списке один элемент
65          if (begin.getNext() == null) {
66              begin = null;
```

```
            return;
        }
        //элементов в списке два и более
        NodePerson current = begin;
        while (current.getNext().getNext() != null) {
           current = current.getNext();
        }
        current.setNext(null);
    }
    //удаление элемента по номеру
    void removeNodeTag(int n) {
        //если списка не существует
        if (begin == null) return;
        //в списке один элемент
        if (begin.getNext() == null) {
            begin = null;
            return;
        }
        //движение по списку до конца или до ук. номера
        NodePerson current = begin;
        for(int i = 2;
            current.getNext().getNext() != null && i < n;
            i++) {
            current = current.getNext();
        }
        //удаление
        current.setNext(current.getNext().getNext());
    }
}
```

//Файл Main.java

Main.java | Person.java | NodePerson.java | Applicable.java | UserList.java | Libr

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        // ввод и проверка условий
        do {
            System.out.print("Введите натуральное n: ");
            n = Integer.parseInt(in.nextLine());
        }
        while (Library.conditionVerify(n) == false);
        //создание списка
        AdvancedList list = new AdvancedList(n);
        System.out.println("Исходный список:");
```

```
17          System.out.println(list.toString());
18          System.out.print("Введите пол для выборки: ");
19          char gender = in.nextLine().charAt(0);
20          System.out.println("Результат выбора");
21          System.out.println(list.resultChoise(gender));
22          System.out.println("Дабавим элемент в начало " +
23                             "списка");
24          list.addNodeBegin();
25          System.out.println("Результат");
26          System.out.println(list.toString());
27          System.out.println("Дабавим элемент в конец");
28          list.addNodeEnd();
29          System.out.println("Результат");
30          System.out.println(list.toString());
31          System.out.println("Дабавим на позицию 2 элемент");
32          list.addNodeTag(2);
33          System.out.println("Результат");
34          System.out.println(list.toString());
35          System.out.println("Удалим элемент из начала " +
36                             "списка");
37          list.removeNodeBegin();
38          System.out.println("Результат");
39          System.out.println(list.toString());
40          System.out.println("Удалим последний элемент в " +
41                             "списке");
42          list.removeNodeEnd();
43          System.out.println("Результат");
44          System.out.println(list.toString());;
45          System.out.println("Удалим 2-ой элемент в " +
46                             "списке");
47          list.removeNodeTag(2);
48          System.out.println("Результат");
49          System.out.println(list.toString());
50      }
51  }
```

Результат работы программы:

```
Введите натуральное n: 3
Ввод 1-ой записи:
        Фамилия:
          Иванов
        Имя:
          Иван
        Отчевство:
          Иванович
```

```
        Отчевство:
          Иванович
        Возраст:
          33
        Пол:
          м
Ввод 2-ой записи:
        Фамилия:
          Фурцева
        Имя:
          Аделия
        Отчевство:
          Петровна
        Возраст:
          66
        Пол:
          ж
Ввод 3-ой записи:
        Фамилия:
          Петров
        Имя:
          Петр
        Отчевство:
          Петрович
        Возраст:
          45
        Пол:
          м
Исходный список:
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м

Введите пол для выборки: м
Результат выбора
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Петров П.П.   возраст: 45, пол: м

Дабавим элемент в начало списка
        Фамилия:
          Чуйко
        Имя:
```

```
          Сидор
        Отчевство:
          Федорович
        Возраст:
          43
        Пол:
          м
Результат
ФИО Чуйко с.Ф.   возраст: 43, пол: м
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м

Дабавим элемент в конец
        Фамилия:
          Сидоров
        Имя:
          Сидор
        Отчевство:
          Сидорович
        Возраст:
          15
        Пол:
          м
Результат
ФИО Чуйко с.Ф.   возраст: 43, пол: м
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м
ФИО Сидоров С.С.   возраст: 15, пол: м

Дабавим на позицию 2 элемент
        Фамилия:
          Двушкин
        Имя:
          Марьян
        Отчевство:
          Эдуардович
        Возраст:
          37
        Пол:
          м
```

```
Результат
ФИО Чуйко с.Ф.   возраст: 43, пол: м
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м
ФИО Сидоров С.С.   возраст: 15, пол: м

Дабавим на позицию 2 элемент
        Фамилия:
          Двушкин
        Имя:
          Марьян
        Отчевство:
          Эдуардович
        Возраст:
          37
        Пол:
          м
Результат
ФИО Чуйко с.Ф.   возраст: 43, пол: м
ФИО Двушкин М.Э.   возраст: 37, пол: м
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м
ФИО Сидоров С.С.   возраст: 15, пол: м

Удалим элемент из начала списка
Результат
ФИО Двушкин М.Э.   возраст: 37, пол: м
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м
ФИО Сидоров С.С.   возраст: 15, пол: м

Удалим последний элемент в списке
Результат
ФИО Двушкин М.Э.   возраст: 37, пол: м
ФИО Иванов В.И.   возраст: 33, пол: м
ФИО Фурцева А.П.   возраст: 66, пол: ж
ФИО Петров П.П.   возраст: 45, пол: м

Удалим 2-ой элемент в списке
```

```
Результат
ФИО Двушкин М.Э.    возраст: 37, пол: м
ФИО Фурцева А.П.    возраст: 66, пол: ж
ФИО Петров П.П.    возраст: 45, пол: м
```

4.1.1.3. Индивидуальные задания

4.1.1.3.1. Часть 1

1. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***четвертого поля*** со введенным с клавиатуры значением;
2. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***пятого поля*** со введенным с клавиатуры значением.
3. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***шестого поля*** со введенным с клавиатуры значением.
4. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***седьмого поля*** со введенным с клавиатуры значением.
5. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***первого поля*** со введенным с клавиатуры значением.
6. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***второго поля*** со введенным с клавиатуры значением.
7. Написать метод, ***добавляющий один элемент*** списка (в середину или конец) после первого по порядку элемента, у которого совпадает значение ***третьего поля*** со введенным с клавиатуры значением.

4.1.1.3.2. Часть 2

1. написать метод, ***удаляющий все элементы*** списка (из середины и конца), у которого совпадает значение **первого поля** со введенным с клавиатуры значением;
2. написать метод, ***удаляющий все элементы*** списка (из середины или конца), у которого совпадает значение ***второго поля*** со введенным с клавиатуры значением;
3. написать метод, ***удаляющий все элементы*** списка (из середины или конца), у которого совпадает значение ***третьего поля*** со введенным с клавиатуры значением;
4. написать метод, ***удаляющий все элементы*** списка (из середины или конца), у которого совпадает значение ***четвертого поля*** со введенным с клавиатуры значением;
5. написать метод, ***удаляющий все элементы*** списка (из середины или конца), у которого совпадает значение ***пятого поля*** со введенным с клавиатуры значением;
6. написать метод, ***удаляющий все элементы*** списка (из середины или конца), у которого совпадает значение ***шестого поля*** со введенным с клавиатуры значением;
7. написать метод, ***удаляющий все элементы*** списка (из середины или конца), у которого совпадает значение ***седьмого поля*** со введенным с клавиатуры значением.

4.1.2. Дальнейшее расширение функциональности проекта наследованием. Редактирование полей списков. Обработка исключений

Обобщенная формулировка задания. В соответствии с предыдущим заданием из пункта «4.1.1. Разработка класса, содержащего методы добавления/удаления новых узлов списка» для выполнения текущего задания проект уже должен содержать семь файлов (Рисунок 8):

1. `Main.java`;
2. `ИмяКласса.java`;
3. `NodeИмяКласса.java`;
4. `Applicable.java`;
5. `UserList.java`;
6. `Library.java`;
7. `AdvancedList.java`.

В соответствии с текущим заданием упомянутый проект из предыдущего задания дополняется ***восьмым файлом*** `ExpandedList.java` (Рисунок 15), содержащем публичный класс

`ExpandedList`, расширяющий класс `AdvancedList` и содержащим публичный класс `ExpandedList` с *экземплярными* методами (по одному) из *трех списков* раздела «Индивидуальные задания» данной темы.

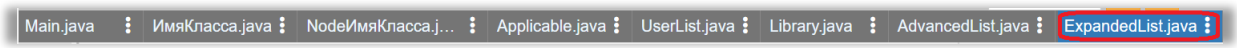


Рисунок 15 – Список файлов нового проекта

Замечания:

- *при необходимости класс `ИмяКласса` (файл `ИмяКласса.java`) дополняется необходимыми сеттерами;*
- *при разработке методов нового класса `ExpandedList`, где это необходимо предусмотреть обработку исключительных ситуаций, которые могут возникнуть по вине пользователя;*
- *при демонстрации программы необходимо последовательно вызвать все методы в `main()`;*
- *метод `toString()` переопределяется во всех классах.*

Пример задания. На основе предыдущего проекта, расширяющего функциональность односвязного списка, состоящего из узлов, содержащих информационную часть в соответствии с предметной областью «Человек» (`Person`), используя наследование продолжить расширять функциональность.

Для этого необходимо создать класс `ExpandedList` наследующем класс `AdvancedList` из предыдущего проекта. Новый класс `ExpandedList` будет содержать метод исправления в объекте типа `Person` в сформированном односвязном списке, значение поля `age`. Объект (узел или нода) списка, в котором выполняются исправления определяется по заданному номеру.

Замечание.

Для выполнения примера задания необходимо в исходном проекте класс `Person` дополнить сеттером `setAge()`.

4.1.2.1. Пример выполнения задания

```
//Файл Person.java изменения выделены цветом
```

Main.java | Person.java | NodePerson.java | Applicable.java | UserList.java | Li

```java
public class Person {
    //свойства
    private String firstName, secondName, lastName;
    private int age;
    private char gender;
    //методы
    public Person(String firstName, String secondName,
           String lastName, int age, char gender) {
        this.firstName = firstName;
        this.secondName = secondName;
        this.lastName = lastName;
        this.age = age;
        this.gender = gender;
    }
    public String getFirst()  {return firstName;}
    public String getSecond() {return secondName;}
    public String getLast()   {return lastName;}
    public int getAge()       {return age;}
    public char getGender()   {return gender;}
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "ФИО " + this.getLast() +
              " " + this.getFirst().charAt(0)+ "." +
               this.getSecond().charAt(0)+ "." +
               "   возраст: " + this.getAge() + ", пол: " +
               this.getGender();
    }
}
```

```
//Файлы: NodePerson.java,
//       Applicable.java
//       UserList.java
//       Library.java
//       AdvancedList.java
//остаются из предыдущего проекта без изменений

//Файл ExpandedList.java
```

Applicable.java | UserList.java | Library.java | AdvancedList.java | ExpandedList.java

```java
import java.util.Scanner;

public class ExpandedList extends AdvancedList {
    //конструктор
    ExpandedList(int n) {
        super(n);
    }
    public void editAge(int number) {
        NodePerson current = begin;
        for(int i = 1; i < number; i++) {
            //обработка исключения - выход current-а за
            //пределы списка из-за number
            try {
                current = current.getNext();
            }
            catch (NullPointerException ex) {
                System.out.println("Выход за пределы списка");
                return;
            }
        }
        Scanner in = new Scanner(System.in);
        System.out.println("\tНовый возраст: ");
        System.out.print("\t  ");
        int age;
        try {
            age = Integer.parseInt(in.nextLine());
        }
        catch (NumberFormatException e) {
            age = -1;
        }
        in.close();
        current.getPerson().setAge(age);
    }
}
```

//Файл Main.java

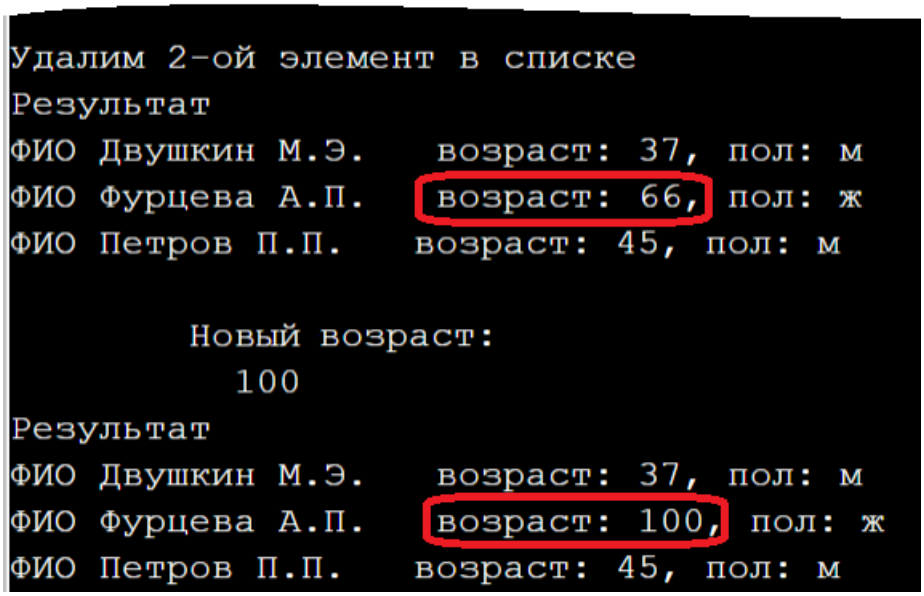
Main.java | Person.java | NodePerson.java | Applicable.java | UserList.java | Lib

```java
import java.util.Scanner;

public class Main
{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n;
        // ввод и проверка условий
        do {
            System.out.print("Введите натуральное n: ");
            n = Integer.parseInt(in.nextLine());
        }
        while (Library.conditionVerify(n) == false);
        //создание списка
        ExpandedList list = new ExpandedList(n);
        System.out.println("Исходный список:");
        System.out.println(list.toString());
```

```
18          System.out.print("Введите пол для выборки: ");
19          char gender = in.nextLine().charAt(0);
20          System.out.println("Результат выбора");
21          System.out.println(list.resultChoise(gender));
22          System.out.println("Дабавим элемент в начало " +
23                             "списка");
24          list.addNodeBegin();
25          System.out.println("Результат");
26          System.out.println(list.toString());
27          System.out.println("Дабавим элемент в конец");
28          list.addNodeEnd();
29          System.out.println("Результат");
30          System.out.println(list.toString());
31          System.out.println("Дабавим на позицию 2 элемент");
32          list.addNodeTag(2);
33          System.out.println("Результат");
34          System.out.println(list.toString());
35          System.out.println("Удалим элемент из начала " +
36                             "списка");
37          list.removeNodeBegin();
38          System.out.println("Результат");
39          System.out.println(list.toString());
40          System.out.println("Удалим последний элемент в " +
41                             "списке");
42          list.removeNodeEnd();
43          System.out.println("Результат");
44          System.out.println(list.toString());;
45          System.out.println("Удалим 2-ой элемент в " +
46                             "списке");
47          list.removeNodeTag(2);
48          System.out.println("Результат");
49          System.out.println(list.toString());
50          list.editAge(2);
51          System.out.println("Результат");
52          System.out.println(list.toString());
53      }
54  }
```

Результат работы программы в целом совпадает с результатом предыдущего проекта пункта «4.1.1. Разработка класса, содержащего методы добавления/удаления новых узлов списка». Основным отличием является дополнение результатов вывода предыдущего проекта выводом списка оставшихся после редактирования записей с измененным годом во второй по номеру записи:



4.1.2.2. Индивидуальные задания

4.1.2.2.1. Часть 1

Создание метода, выполняющего действия согласно заданию:

1. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***первого*** поля и ***изменяющим это значение*** на новое, введенное с клавиатуры.
2. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***второго*** поля и ***изменяющим это значение*** на новое, введенное с клавиатуры.
3. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***третьего*** поля и ***изменяющим это значение*** на новое, введенное с клавиатуры.
4. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***четвертого*** поля и ***изменяющим это значение*** на новое, введенное с клавиатуры.
5. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***пятого*** поля и ***изменяющим это значение*** на новое, введенное с клавиатуры.
6. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***первого*** поля и изменить ***значение второго поля этого элемента*** на новое, введенное с клавиатуры.
7. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***третьего*** поля и изменить

значение <u>второго</u> поля этого элемента на новое, введенное с клавиатуры.

8. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>четвертого</u>*** поля и изменить ***значение <u>второго</u> поля этого элемента*** на новое, введенное с клавиатуры.
9. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>пятого</u>*** поля и изменить ***значение <u>второго</u> поля этого элемента*** на новое, введенное с клавиатуры.
10. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>второго</u>*** поля и изменить ***значение <u>третьего</u> поля этого элемента*** на новое, введенное с клавиатуры.
11. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>второго</u>*** поля и изменить ***значение <u>четвертого</u> поля этого элемента*** на новое, введенное с клавиатуры.
12. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>второго</u>*** поля и изменить ***значение <u>пятого</u> поля этого элемента*** на новое, введенное с клавиатуры.
13. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>третьего</u>*** поля и изменить ***значение <u>четвертого</u> поля этого элемента*** на новое, введенное с клавиатуры.
14. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>третьего</u>*** поля и изменить ***значение <u>пятого</u> поля этого элемента*** на новое, введенное с клавиатуры.
15. Написать метод, находящий в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***<u>четвертого</u>*** поля и изменить ***значение <u>пятого</u> поля этого элемента*** на новое, введенное с клавиатуры.

4.1.2.2.2. Часть 2

Разработка метода, заменяющий только информационную часть узла (ноды) списка (объект `ИмяКласса`):

1. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***первого*** поля и заменить

только информационную часть (в примере объект класса `Person`) найденного узла.

2. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***второго*** поля и заменить только информационную часть (в примере объект класса `Person`) найденного узла.
3. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***третьего*** поля и заменить только информационную часть (в примере объект класса `Person`) найденного узла.
4. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***четвертого*** поля и заменить только информационную часть (в примере объект класса `Person`) найденного узла.
5. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***пятого*** поля и заменить только информационную часть (в примере объект класса `Person`) найденного узла.

4.1.2.2.3. Часть 3

Разработка метода, заменяющего весь узел списка целиком (объект `NodeИмяКласса`):

1. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***первого*** поля и заменить найденный узел целиком.
2. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***второго*** поля и заменить найденный узел целиком.
3. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***третьего*** поля и заменить найденный узел целиком.
4. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***четвертого*** поля и заменить найденный узел целиком.
5. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением ***пятого*** поля и заменить найденный узел целиком.

4.2. Динамическая диспетчеризация методов

4.2.1. Двусвязный список из объектов разных классов, имеющих общий пользовательский класс-родитель

Обобщенная формулировка задания. В соответствии с индивидуальным заданием на основе базового класса, представляющего собой предметную область, необходимо создать программу для связывания объектов нескольких различных классов в двусвязный список. Вывести список на экран.

Общие требования к выполнению задания:

- каждый класс должен быть помещен в отдельный файл с именем `ИмяКласса.java`;
- класс `BaseInform`, описывающий предметную область, должен иметь ***не менее пяти*** свойств, содержащих общую информацию;
- базовый класс в иерархии, должен на правах агрегации содержать объект класса `BaseInform`;
- все классы наследники от базового класса, должны иметь дополнительно к базовой информации ***не менее трех*** свойств, описывающих объект класса наследника;
- список должны формироваться из объектов не менее чем трех классов, согласно индивидуальному заданию.

Замечание.
Метод `toString()` переопределяется во всех классах.

Пример задания. На основе предметной области «Родитель» (`Parent`) создать с помощью вызова метода, связывающего список, через явный вызов конструкторов классов наследников «Совершеннолетний ребенок» (`AdultChild`) и «Несовершеннолетний ребенок» (`UndergradeChild`). Вывести на экран содержимое списка.

Диаграмма классов будет иметь следующий вид, представленный на рисунке (Рисунок 16).

Исходя из того, что восходящие преобразования будут осуществляться автоматически, то создав объекты разных классов `AdultChild` и `UndergradeChild`, являющихся наследниками класса `Parent`, эти созданные объекты можно будет связать в двусвязный список.

Порядок связывания (очередность типов связываемых объектов) диктуется исключительно требованиями разработчика.

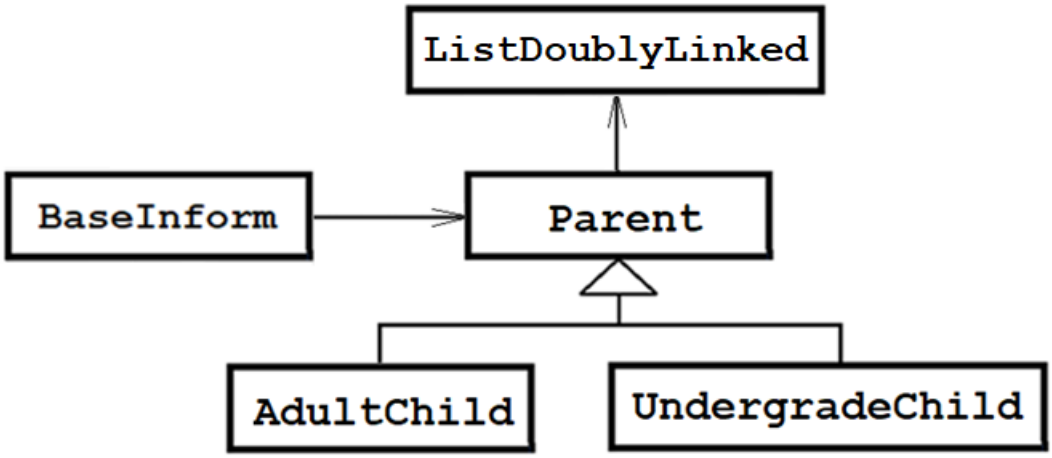


Рисунок 16 – Диаграмма классов, объединенных классом, описывающим предметную область

4.2.1.1. Пример выполнения задания

//файл BaseInform.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j… | AdultChild.j

```
1  public class BaseInform {
2      protected String name;
3      protected char gender;
4      protected int age;
5      //только контсруктор
6      BaseInform(String name, char gender, int age) {
7          this.name = name;
8          this.gender = gender;
9          this.age = age;
10     }
11     @Override
12     public String toString() {
13         return "name: " + name + ",\tage: " + age + ", ";
14     }
15 }
```

//файл Parent.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j…

```
1  public class Parent {
2      protected BaseInform inf;
3      //раздел указателей для двусвязного списка
4      protected Parent next = null;
5      protected Parent prev = null;
```

```
    //методы
    Parent(BaseInform inf) {
        this.inf = inf;
    }
    Parent getNext() {
        return next;
    }
    void setNext(Parent s) {
        next = s;
    }
    void setPrev(Parent s) {
        prev = s;
    }
    @Override
    public String toString() {
       return inf.toString();
    }
}
```

//Файл ListDoublyLinked.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j...

```
public class ListDoublyLinked {
    private Parent begin;
    //конструктор
    public ListDoublyLinked(Parent[] a) {
        begin = a[0];
        for(int i = 1; i < a.length; i++) {
            addNodeEnd(a[i]);
        }
    }
    //метод добавления элемента списка в конец
    void addNodeEnd(Parent a) {
        Parent current = begin;
        while (current.getNext() != null) {
           current = current.getNext();
        }
        current.setNext(a);
        current.getNext().setPrev(current);
    }
```

```
    @Override
    public String toString() {
        String result = "";
        Parent current = begin;
        while (current != null) {
            result = result
                    + current.toString()
                    + "\n";
            current = current.getNext();
        }
        return result;
    }
}
```

//Файл AdultChild.java

ent.java | ListDoublyLinked.j... | AdultChild.java | UnderageChild.java

```
class AdultChild extends Parent {
    private float income;
    //методы
    AdultChild(BaseInform inf, float income) {
        super(inf);
        this.income = income;
    }
    @Override
    public String toString() {
        return "Adult: \t\t" + inf.toString() +
                "\tincome: " + income;
    }
}
```

//Файл UnderageChild.java

ent.java | ListDoublyLinked.j... | AdultChild.java | UnderageChild.java

```
class UnderageChild extends Parent {
    private int iq;
    //методы
    UnderageChild(BaseInform inf, int iq) {
        super(inf);
        this.iq = iq;
    }
```

```
    @Override
    public String toString() {
        return "Underage: \t" + inf.toString()
                + "\tIQ: " + iq;
    }
}
```

//файл Main.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j… | AdultChild.java | UnderageCh

```
public class Main
{
    public static void main(String[] args) {
        Parent[] array =
            {new AdultChild(new BaseInform("Sam",'m', 28), 1200),
             new UnderageChild(new BaseInform("John", 'm', 12), 120)};
        ListDoublyLinked list = new ListDoublyLinked(array);
        System.out.println(list.toString());
    }
}
```

Результат выполнения программы:

input

```
Adult:          name: Sam,      age: 28,        income: 1200.0
Underage:       name: John,     age: 12,        IQ: 120
```

4.2.1.2. Индивидуальные задания

1. Создать базовый класс (предметная область) «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка».
2. Создать базовый класс (предметная область) «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль».
3. Создать базовый класс (предметная область) «Учащийся» и производные классы «Школьник», «Учащийся ГПТУ» и «Студент».
4. Создать базовый класс (предметная область) «Музыкальный инструмент» и производные классы «Ударный», «Струнный», «Духовой».
5. Создать базовый класс (предметная область) «Работник фирмы» и производные классы «Менеджер», «Администратор», «Программист».
6. Создать базовый класс (предметная область) «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай».

7. Создать базовый класс (предметная область) «Садовое дерево» и производные классы «Яблоня», «Вишня», «Груша».
8. Создать базовый класс (предметная область) «Единица хранения в библиотеке» и производные классы «Художественная книга», «Научная книга», «Журнал».
9. Создать базовый класс (предметная область) «Автомобиль» и производные классы «Легковой автомобиль», «Грузовой автомобиль», «Вездеход».
10. Создать базовый класс (предметная область) «Плавсредство» и производные классы «Корабль», «Баржа», «Яхта».
11. Создать базовый класс (предметная область) «Электростанция» и производные классы «Теплоэлектростанция», «Гидроэлектростанция», «Атомная станция».
12. Создать базовый класс (предметная область) «Городской транспорт» и производные классы «Троллейбус», «Автобус», «Трамвай».
13. Создать базовый класс (предметная область) «Коммунальная техника» и производные классы «Уборочная техника», «Мусоровоз», «Автомобиль-разбрасыватель реагентов».
14. Создать базовый класс (предметная область) «Торговая сеть» и производные классы «Гипермаркет», «Специализированные магазин», «Рынок».
15. Создать базовый класс (предметная область) «Школа» и производные классы «Учитель», «Школьник», «Повар».
17. Создать базовый класс (предметная область) «Рабочий» и производные классы «Слесарь», «Электрик», «Монтажник-высотник».

4.2.2. Расширение функциональности проекта с помощью внутренних и анонимных классов

Обобщенная формулировка задания. В проект, разработанный в предыдущем пункте «4.2.1. Двусвязный список из объектов разных классов, имеющих общий пользовательский класс-родитель» внести изменения:

- добавить необходимые геттеры в классы, входящие в состав проекта;
- для класса `ПредметнОбласть` с помощью внутренних классов переопределить метод `toString()` несколько раз:
  - для вывода на экран в ***прямом*** порядке полной информации об элементах списка;
  - для вывода на экран в ***обратном*** порядке полной информации об элементах списка;

    - для вывода на экран в ***прямом*** порядке заданного в ***части 1 индивидуального задания*** поля класса `BaseInform`;
    - для вывода на экран в ***обратном*** порядке заданного в ***части 2 индивидуального задания*** поля класса `BaseInform`;
  - в классах ***наследниках*** с помощью внутренних классов ***добавить по одному дополнительному*** полю и переопределить метод `toString()` таким образом, чтобы значение нового поля выводилось на экран;
  - с помощью анонимных классов в `main()` добавить дополнительное поле с заданным значением в объект корректируемого класса и вывести на экран сокращенную информацию об объекте с учетом дополнительного значения.

Общие требования к выполнению задания:

- ***класс*** `BaseInform` должен содержать не менее ***трех*** свойств, описывающих предметную область;
- ***класс*** с именем `ПредметнОбласть` должен быть связан отношением агрегации с классом `BaseInform`;
- ***класс*** с именем `ListDoublyLinked` должен содержать:
  - конструктор;
  - метод формирующий список добавлением элемента в конец;
  - ***четыре вложенных*** класса, содержащих ***по одному*** переопределенному методу `toString()` вывода на экран информации в соответствии с обобщенным заданием (см. выше);
- ***производные классы***, уточняющие класс `ПредметнОбласть` должны:
  - быть помещен в отдельные файлы с именем `ИмяКласса.java`;
  - должны иметь не менее ***трех*** свойств, уточняющих предметную область;
- в методе `main()`:
  - на *первом* этапе список должны формироваться из объектов ***не менее чем трех различных классов***, согласно индивидуальному заданию;
  - на *втором* этапе должен выводиться список ***в прямом и обратном*** порядке всех свойств дочерних классов (полные записи);
  - на *третьем* этапе должен выводиться список ***в прямом и обратном*** порядке ***одного*** из свойства класса `BaseInform`, указанном в ***индивидуальном задании***;

- на *четвертом* этапе в `main()` с помощью ***анонимных*** классов должны переопределяться методы `toString()`, позволяющие вывести сокращенную запись свойств;
- на *пятом* этапе в список добавляются объекты анонимных классов;
- на *шестом* этапе выводится на экран полный список всех объектов.

Пример задания. На основе предметной области «Родитель» (`Parent`) создать с помощью вызова метода, связывающего список, через явный вызов конструкторов классов наследников «Совершеннолетний ребенок» (`AdultChild`) и «Несовершеннолетний ребенок» (`UnderageChild`). Вывести на экран содержимое списка.

4.2.2.1. Пример выполнения задания

//файл BaseInform.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j… | AdultChild.j

```
1  public class BaseInform {
2      protected String name;
3      protected char gender;
4      protected int age;
5      //только контсруктор
6      BaseInform(String name, char gender, int age) {
7          this.name = name;
8          this.gender = gender;
9          this.age = age;
10     }
11     String getName() {
12         return name;
13     }
14     @Override
15     public String toString() {
16         return "name: " + name + ",\tage: " + age + ", ";
17     }
18 }
```

//файл Parent.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j…

```
1  public class Parent {
2      protected BaseInform inf;
3      //раздел указателей для двусвязного списка
```

```
4       protected Parent next = null;
5       protected Parent prev = null;
6       //методы
7       Parent(BaseInform inf) {
8           this.inf = inf;
9       }
10      Parent getNext() {
11          return next;
12      }
13      Parent getPrev() {
14          return prev;
15      }
16      BaseInform getBaseInform() {
17          return inf;
18      }
19      void setNext(Parent s) {
20          next = s;
21      }
22      void setPrev(Parent s) {
23          prev = s;
24      }
25      @Override
26      public String toString() {
27         return inf.toString();
28      }
29  }
```

//файл ListDoublyLinked.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j... | Ad

```
1   public class ListDoublyLinked {
2       private Parent begin;
3       //конструктор
4       public ListDoublyLinked(Parent[] a) {
5           begin = a[0];
6           for(int i = 1; i < a.length; i++) {
7               addNodeEnd(a[i]);
8           }
9       }
10      //метод добавления элемента списка в конец
11      public void addNodeEnd(Parent a) {
12          Parent current = begin;
```

```
13          while (current.getNext() != null) {
14              current = current.getNext();
15          }
16          current.setNext(a);
17          current.getNext().setPrev(current);
18      }
19      //вложенный класс DirectPrintFullDetails
20      public class DirectPrintFullDetails {
21          @Override
22          public String toString() {
23              String result = "";
24              Parent current = begin;
25              while (current != null) {
26                  result = result
27                          + current.toString()
28                          + "\n";
29                  current = current.getNext();
30              }
31              return result;
32          }
33      }
34      //вложенный класс ReversePrintFullDetails
35      public class ReversePrintFullDetails {
36          @Override
37          public String toString() {
38              Parent current = begin;
39              //вспомогательный проход до конца
40              //списка
41              while (current.getNext() != null) {
42                  current = current.getNext();
43              }
44              String result = "";
45              while (current != null) {
46                  result = result
47                          + current.toString()
48                          + "\n";
49                  current = current.getPrev();
50              }
51              return result;
52          }
53      }
```

```
54      //вложенный класс NameDirectOrder
55      public class NameDirectOrder {
56          @Override
57          public String toString() {
58              String result = "";
59              Parent current = begin;
60              while (current != null) {
61                  result = result
62                          + current.getBaseInform()
63                                  .getName()
64                                  .toString()
65                          + "\n";
66                  current = current.getNext();
67              }
68              return result;
69          }
70      }
71      //вложенный класс NameReverseOrder
72      public class NameReverseOrder {
73          @Override
74          public String toString() {
75              Parent current = begin;
76              //вспомогательный проход до конца
77              //списка
78              while (current.getNext() != null) {
79                  current = current.getNext();
80              }
81              String result = "";
82              while (current != null) {
83                  result = result
84                          + current.getBaseInform()
85                                  .getName()
86                                  .toString()
87                          + "\n";
88                  current = current.getPrev();
89              }
90              return result;
91          }
92      }
93  }
```

//Файл AdultChild.java

ava | ListDoublyLinked.j... | AdultChild.java | UnderageChild.java

```
1  public class AdultChild extends Parent {
2      private double income;
3      //методы
4      AdultChild(BaseInform inf, double income) {
5          super(inf);
6          this.income = income;
7      }
8      double getIncome() {
9          return income;
10     }
11     @Override
12     public String toString() {
13         return "Adult: \t\t" + inf.toString() +
14                 "\tincome: " + income;
15     }
16 }
```

//Файл UnderageChild.java

t.java | ListDoublyLinked.j... | AdultChild.java | UnderageChild.java

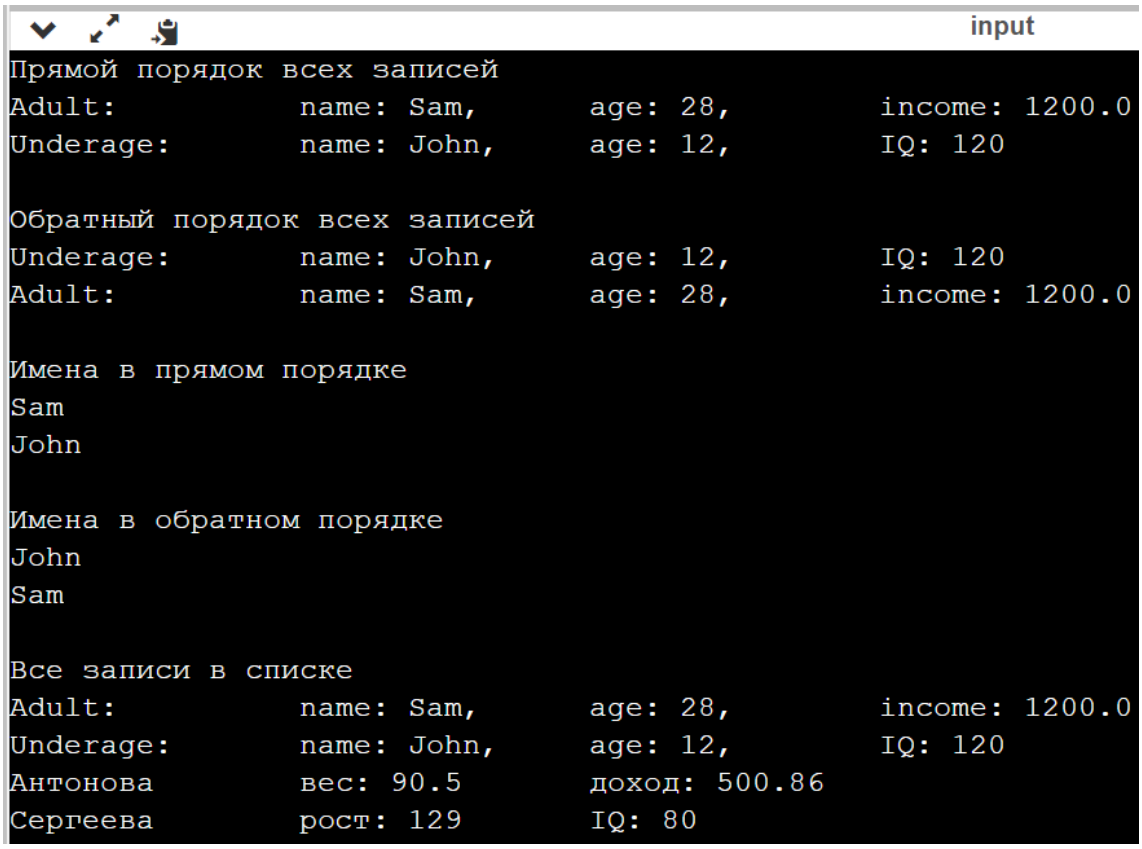
```
1  class UnderageChild extends Parent {
2      private int iq;
3      //методы
4      UnderageChild(BaseInform inf, int iq) {
5          super(inf);
6          this.iq = iq;
7      }
8      int getIq() {
9          return iq;
10     }
11     @Override
12     public String toString() {
13         return "Underage: \t" + inf.toString()
14                 + "\tIQ: " + iq;
15     }
16 }
```

//Файл Main.java

Main.java | BaseInform.java | Parent.java | ListDoublyLinked.j... | AdultChild.java | UnderageChild.java

```
1  public class Main
2  {
3      public static void main(String[] args) {
4          Parent[] array =
5              {new AdultChild(new BaseInform("Sam",'m', 28), 1200),
6               new UnderageChild(new BaseInform("John", 'm', 12), 120)};
7          //1. Формирование списка из стандартных записей
8          ListDoublyLinked list = new ListDoublyLinked(array);
9          //2. Вывод на экран полных записей
10         System.out.println("Прямой порядок всех записей");
11         System.out.println(list.new DirectPrintFullDetails().toString());
12         System.out.println("Обратный порядок всех записей");
13         System.out.println(list.new ReversePrintFullDetails().toString());
14         //3. Вывод на экран сокращенных записей
15         System.out.println("Имена в прямом порядке");
16         System.out.println(list.new NameDirectOrder().toString());
17         System.out.println("Имена в обратном порядке");
18         System.out.println(list.new NameReverseOrder().toString());
19         //4 и 5 этапы. Анонимные классы и добавление их объектов в список
20         //переопределение первого класса наследника
21         Parent a = new AdultChild(new BaseInform("Антонова",'ж', 35), 500.86) {
22                         //анонимный класс
23                         double weight = 90.5;
24                         @Override
25                         public String toString() {
26                             return this.getBaseInform().getName()
27                                     + "\tвес: " + weight
28                                     + "\tдоход: "
29                                     + this.getIncome();
30                         }
31                     }; //окончание анонимного класса
32         //добавление модифицированного объекта в список
33         list.addNodeEnd(a);
34         //переопределение второго класса наследника
35         a = new UnderageChild(new BaseInform("Сергеева",'ж', 5), 80) {
36                         //анонимный класс
37                         int height = 129;
38                         @Override
39                         public String toString() {
40                             return this.getBaseInform().getName()
41                                     + "\tрост: " + height
42                                     + "\tIQ: "
43                                     + this.getIq();
44                         }
45                     }; //окончание анонимного класса
46         //добавление модифицированного объекта в список
47         list.addNodeEnd(a);
48         //6 этап. Вывод на экран всех объектов
49         System.out.println("Все записи в списке");
50         System.out.println(list.new DirectPrintFullDetails().toString());
51     }
52 }
```

Результат выполнения программы:



4.2.2.2. Индивидуальные задания

4.2.2.2.1. Часть 1

Создание списка из объектов классов-наследников. См. задания из пункта «4.2.1.2. Индивидуальные задания».

4.2.2.2.2. Часть 2

Создать внутренние классы, с переопределенным методом `toString()`, и с его помощью:

- вывести полный список ***всех*** полей объектов списка в прямом и обратном порядке;
- вывести полный список ***нескольких*** полей на экран в прямом и обратном порядке:
    1. первое свойство класса `BaseInform`;
    2. второе свойство класса `BaseInform`;

3. третье свойство класса `BaseInform`;
4. первое и второе свойства `BaseInform`;
5. первое и третье свойства `BaseInform`;
6. второе и третье свойства `BaseInform`.

4.2.2.2.3. Часть 3

Анонимные классы. С помощью анонимных классов переопределить сокращенный вывод с помощью метода `toString()` свойств ***классов-наследников***:

1. первое свойство каждого свойства класса-наследника;
2. второе свойство каждого свойства класса-наследника;
3. третье свойство каждого свойства класса-наследника;
4. первое и второе каждого свойства класса-наследника;
5. первое и третье каждого свойства класса-наследника;
6. второе и третье каждого свойства класса-наследника.

4.2.3. Двусвязный список из объектов разных классов, не имеющих в иерархии объединяющего класса, описывающего предметную область

Обобщенная формулировка задания. В соответствии с индивидуальным заданием необходимо создать программу для связывания объектов нескольких различных классов в двусвязный список. Вывести список на экран.

Общие требования к выполнению задания:

- каждый класс должен быть помещен в отдельный файл с именем `ИмяКласса.java`;
- каждый класс, описывающий конкретное понятие должен иметь ***не менее пяти*** свойств;
- класс `Node` в качестве свойств должен иметь только ссылки на следующий и предыдущий элемент списка;
- список должны формироваться из объектов не менее чем ***пяти*** классов, согласно индивидуальному заданию.

Замечание.
Метод `toString()` переопределяется во всех классах.

Пример задания. Создать двусвязный список объектов различных классов с помощью вызова метода, связывающего список, через явный

вызов конструкторов связываемых классов. Вывести на экран содержимое списка. Связываемые классы: военнослужащий (`Soldier`), яблоня (`AppleTree`), автомобиль (`Car`).

Диаграмма классов будет иметь следующий вид (Рисунок 17):

Исходя из того, что восходящие преобразования будут осуществляться автоматически, то создав объекты разных классов `Soldier`, `AppleTree` и `Car`, являющихся наследниками класса `Object`, эти объекты можно будет связать в двусвязный список.

Структура проекта имеет вид, приведенный на рисунке (Рисунок 18).

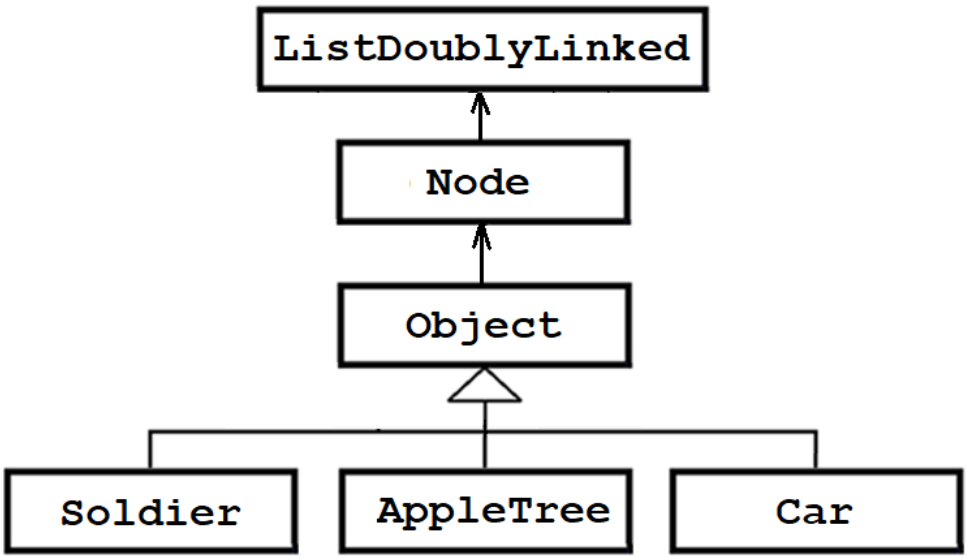


Рисунок 17 – Иерархия связываемых в двусвязный список классов

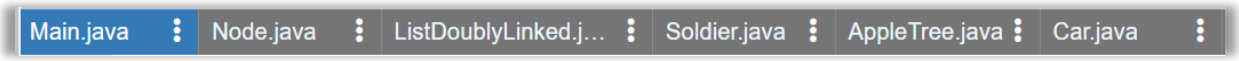


Рисунок 18 – Структура проекта

4.2.3.1. Пример выполнения задания

```
//файл Node.java
```

Main.java | Node.java | ListDoublyLinked.j... | Soldier.java | Ap

```
1  public class Node {
2      protected Object inf;
3      //раздел указателей для двусвязного списка
4      protected Node next = null;
5      protected Node prev = null;
6      //методы
```

```
    Node(Object inf) {
        this.inf = inf;
    }
    Node getNext() {
        return next;
    }
    void setNext(Node s) {
        next = s;
    }
    void setPrev(Node s) {
        prev = s;
    }
    @Override
    public String toString() {
       return inf.toString();
    }
}
```

//Файл ListDoublyLinked.java

Main.java | Node.java | ListDoublyLinked.j... | Soldier.java | Ap

```
public class ListDoublyLinked {
    private Node begin;
    //конструктор
    public ListDoublyLinked(Object[] a) {
        begin = new Node(a[0]);
        for(int i = 1; i < a.length; i++) {
            addNodeEnd(a[i]);
        }
    }
    //метод добавления элемента списка в конец
    void addNodeEnd(Object a) {
        Node current = begin;
        while (current.getNext() != null) {
           current = current.getNext();
        }
        current.setNext(new Node(a));
        current.getNext().setPrev(current);
    }
    @Override
    public String toString() {
        String result = "";
        Node current = begin;
        while (current != null) {
            result = result
```

```
                    + current.toString()
                    + "\n";
            current = current.getNext();
        }
        return result;
    }
}
```

//Файл Soldier.java

Main.java | Node.java | ListDoublyLinked.j... | Soldier.java | A

```
public class Soldier {
    private String lastName;
    private String militaryRank;
    private double income;
    //методы
    Soldier(String lastName,
            String militaryRank,
            double income) {
        this.lastName = lastName;
        this.militaryRank = militaryRank;
        this.income = income;
    }
    @Override
    public String toString() {
        return "Soldier: \t" + lastName
              + " \tзвание: " + militaryRank
              + " \t\tзарплата: " + income;
    }
}
```

//Файл AppleTree.java

le.java | ListDoublyLinked.j... | Soldier.java | AppleTree.java | Car.jav

```
public class AppleTree {
    private String appleVariety;
    private int ageTree;
    //методы
    AppleTree(String appleVariety, int ageTree) {
        this.appleVariety = appleVariety;
        this.ageTree = ageTree;
    }
    @Override
    public String toString() {
```

```
        return "AppleTree: \t" + appleVariety
                + "\tвозраст дерева: " + ageTree;
    }
}
```

//Файл Car.java

ListDoublyLinked.j... | Soldier.java | AppleTree.java | Car.java

```
public class Car {
    private String carModel;
    private int yearCarManufacture;
    //методы
    Car(String carModel, int yearCarManufacture) {
        this.carModel = carModel;
        this.yearCarManufacture = yearCarManufacture;
    }
    @Override
    public String toString() {
        return "Car: \t\t" + carModel
                + "\t\tгод выпуска: "
                + yearCarManufacture;
    }
}
```

//файл Main.java

Main.java | Node.java | ListDoublyLinked.j... | Soldier.java | AppleTree.java

```
public class Main
{
    public static void main(String[] args) {
        Object[] array =
            {new Soldier("Сидоров", "майор", 2139.97),
             new AppleTree("Белый налив", 10),
             new Car("Бьюик", 1947)};
        ListDoublyLinked list = new ListDoublyLinked(array);
        System.out.println(list.toString());
    }
}
```

Результат работы программы:

```
input
Soldier:        Сидоров         звание: майор           зарпл?та: 2139.97
AppleTree:      Белый налив     возраст дерева: 10
Car:            Бьюик           год выпуска: 1947
```

4.2.3.2. Индивидуальные задания

1. Связать двусвязный список из объектов классов:
    - «Школьник»,
    - «Ударный музыкальный инструмент»,
    - «Менеджер»,
    - «Собака»,
    - «Яблоня».
2. Связать двусвязный список из объектов классов:
    - «Учащийся ГПТУ»,
    - «Струнный музыкальный инструмент»,
    - «Администратор»,
    - «Кошка»,
    - «Вишня».
3. Связать двусвязный список из объектов классов:
    - «Студент»,
    - «Духовой музыкальный инструмент»,
    - «Программист»,
    - «Попугай»,
    - «Груша».
4. Связать двусвязный список из объектов классов:
    - «Мебель»,
    - «Ж/д вагон»,
    - «Компьютер»,
    - «Осветительный прибор»,
    - «Хлебобулочное изделие».
5. Связать двусвязный список из объектов классов:
    - «Конфета»,
    - «Отрасль знаний»,
    - «Проездной билет»,
    - «Принтер»,
    - «Медикамент».
6. Связать двусвязный список из объектов классов:
    - «Преподаватель»,
    - «Ж/д локомотив»,
    - «Кондитерское изделие»,
    - «Фрукт»,
    - «Художественная книга».
7. Связать двусвязный список из объектов классов:
    - «Научная книга»,

- «Учебная дисциплина»,
- «Расходные материалы»,
- «Овощ»,
- «Легковой автомобиль».

8. Связать двусвязный список из объектов классов:
   - «Журнал».
   - «Грузовой автомобиль»,
   - «Мясной полуфабрикат»
   - «Корабль»,
   - «ВУЗ».
9. Связать двусвязный список из объектов классов:
   - «Теплоэлектростанция»,
   - «Вездеход».
   - «Баржа»,
   - «Троллейбус»,
   - «Снегоуборочная техника».
10. Связать двусвязный список из объектов классов:
   - «Яхта»,
   - «Гидроэлектростанция»,
   - «Упаковка влажных салфеток»,
   - «Автобус»,
   - «Рулон пакетов для мусора».
11. Связать двусвязный список из объектов классов:
   - «Атомная станция».
   - «Трамвай»,
   - «Зубная паста»,
   - «Гипермаркет»,
   - «Автомобиль-разбрасыватель реагентов».
12. Связать двусвязный список из объектов классов:
   - «Ветроэлектрогенератор»,
   - «Мыло»,
   - «Мусоровоз»,
   - «Специализированные магазин».
13. Связать двусвязный список из объектов классов:
   - «Солнечная батарея»,
   - «Рынок»,
   - «Кожгалантерейное изделие»
   - «Автомобиль»,
   - «Учитель».
14. Связать двусвязный список из объектов классов:
   - «Школьник»,

- «Велосипед»,
- «Самолет»,
- «Электрик»,
- «Изделие из железобетона».

15. Связать двусвязный список из объектов классов:
    - «Повар»,
    - «Монтажник-высотник»,
    - «Пишущая ручка»,
    - «Гужевая повозка»,
    - «Лифт».
16. Связать двусвязный список из объектов классов:
    - «Слесарь»,
    - «Канцелярский предмет»,
    - «Поезд»,
    - «Телевизор»,
    - «Смартфон».

Тема 5. Создание и использование пакетов. Дженерики

В заданиях данной темы рекомендуется использовать интегрированную среду `JDoodle` с включенной опцией `Advanced Java IDE` (https://www.jdoodle.com/online-java-compiler-ide/) (Рисунок 19).

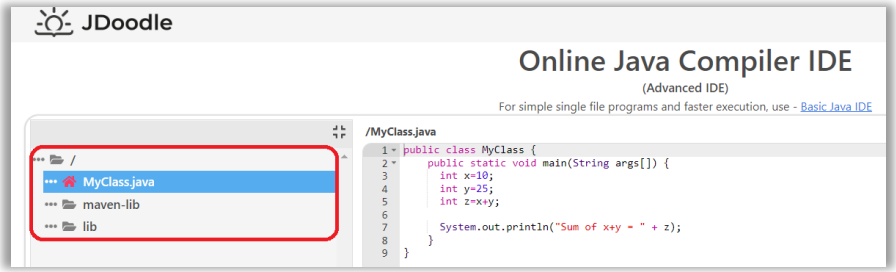


Рисунок 19 – Общий вид окна `JDoodle` с включенной опцией `Advanced Java IDE`

5.1. Параметризованные методы. Использование классов типа record

Задание посвящено созданию многофайлового проекта с использованием ***одного*** пакета, содержащего ***не*** параметризованную класс-

библиотеку `LibraryGenerics`, содержащего ***параметризованные*** статических методы.

Обобщенная формулировка задания. Файл `MyClass.java` с классом `MyClass` будет создан автоматически и будет содержать метод `main()` (собственно программу, использующую разработанные `generic`-и).

В соответствии с индивидуальным заданием следует:

- создать класс (`record`), представляющий предметную область из списка индивидуальных заданий;
- без наследования, но с использованием агрегации создать три класса уточняющих предметную область в соответствии с индивидуальным заданием, т.е. каждый уточняющий класс должен иметь в качестве одного из свойств объект класса, представляющего предметную область;
- каждый из перечисленных классов (в том числе класс, описывающий предметную область) должен иметь минимум по три свойства;
- должен быть создан класс `LibraryGenerics`, имеющий набор методов;
- в класс `LibraryGenerics` вносятся изменения в соответствии с выполняемой частью задания;
- в `main()` создать массив из минимум шести объектов трех типов (уточняющих классов) в произвольном порядке и используя методы класса `LibraryGenerics` вывести:
    - весь массив на экран;
    - только элементы определенного типа;

Замечание.

В метод `main()` класса `MyClass` также вносятся изменения в соответствии с изменениями в классе `LibraryGenerics` для вызова его методов.

Детализация требований к выполнению задания:

- каждый класс должен быть помещен в отдельный файл с именем `ИмяКласса.java`;
- все файлы с именем `ИмяКласса.java` должны быть помещены в каталог (папку, директорию) с именем `lib`;
- каждый файл из директории `lib` должен начинаться инструкцией `package lib`;
- для всех классов, описывающих и уточняющих предметную область должен быть переопределен метод `toString()`;

- должен быть создан не параметризованный класс `LibraryGenerics`, имеющий параметризованные методы;

Создание пакета средствами IDE `JDoodle`. Создание структуры проекта в каталоге `lib` начинается с нажатия на кнопку  на против данного каталога. После чего появляется меню выбора, и разработчик должен определить, что конкретно он создает (файл или каталог, Рисунок 20).

В данном задании студентам необходимо пользоваться только командой `New File` (Рисунок 20), так как пакет один.

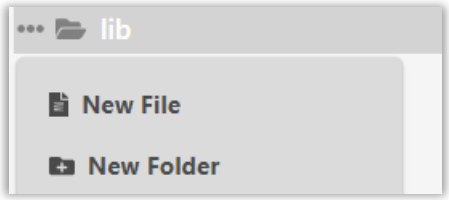


Рисунок 20 - Меню выбора

Пример задания. С использованием предметной области «Основная информация» (`BaseInform`) создать классы, на основе отношения агрегации «Ребенок» (`Child`), «Взрослый» (`Adult`) и «Пенсионер» (`Pensioner`). Вывести на экран содержимое всего массива, а также сделать выборку по типу записей в массиве. Структура проекта приведена на рисунке ниже (Рисунок 21).

Замечание.

Несмотря на порядок создания файлов, они будут упорядочены в алфавитном порядке

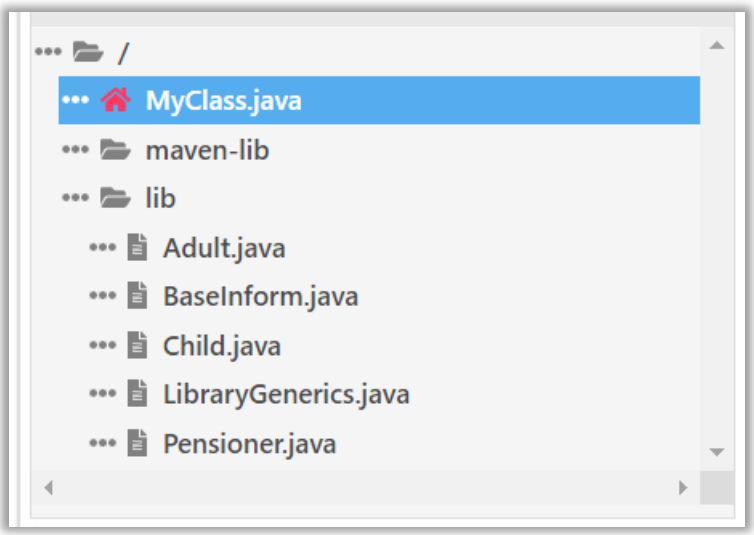


Рисунок 21 – Структура проекта

5.1.1. Пример выполнения задания

```
//пакет lib
```

/lib/Adult.java

```
1  package lib;
2
3  public record Adult (BaseInform inf, double income) {
4      @Override
5      public String toString() {
6          return "Adult: " + inf.name() +
7              " income: " + income + "\n";
8      }
9  }
```

/lib/BaseInform.java

```
1  package lib;
2
3  public record BaseInform (String name,
4                            char gender,
5                            int age) {}
```

/lib/Child.java

```
1  package lib;
2
3  public record Child (BaseInform inf, int iq) {
4      @Override
5      public String toString() {
6          return "Child: " + inf.name() +
7              " IQ: " + iq + "\n";
8      }
9  }
```

/lib/LibraryGenerics.java

```
1  package lib;
2
3  public class LibraryGenerics {
```

```
    public static <T> String choise(T[] a,
                                    String strType) {
        String result = "";
        for(int i = 0; i < a.length; i++) {
            if (a[i].getClass()
                    .getSimpleName()
                    .toUpperCase()
                    .equals(strType.toUpperCase())) {
                result = result + a[i].toString();
            }
        }
        return result;
    }

    public static <T> String toString(T[] a) {
        String str = a[0].toString();
        for(int i = 1; i < a.length; i++) {
            str = str + a[i].toString();

        }
    return str;
    }
}
```

/lib/Pensioner.java

```
package lib;

public record Pensioner(BaseInform inf,
                        double pension) {
    @Override
    public String toString() {
       return "Pensioner: " + inf.name() +
         " pension: " + pension + "\n";
    }
}
```

//точка входа в проект

/MyClass.java

```
import lib.*;

public class MyClass
{
    public static void main(String[] arg) {
```

```
        Object[] a = {
                new Child(new BaseInform("Sidorov", 'm', 15),
                        80),
                new Child(new BaseInform("Ivanov", 'm', 5),
                        120),
                new Adult(new BaseInform("Petrov", 'm', 45),
                        1000000),
                new Pensioner(new BaseInform("Болдырева",
                                'f', 60),
                        500)
        };
        System.out.println(LibraryGenerics.toString(a));
        System.out.println(LibraryGenerics.choise(a, "Pensioner"));
    }
}
```

Результат работы программы:

```
Child: Sidorov IQ: 80
Child: Ivanov IQ: 120
Adult: Petrov income: 1000000.0
Pensioner: Болдырева pension: 500.0

Pensioner: Болдырева pension: 500.0
```

5.1.2. Индивидуальные задания

1. Создать класс предметной области «Транспортное средство» и классы «Автомобиль», «Велосипед», «Повозка», содержащие объект предметной области в качестве свойства.
2. Создать класс предметной области «Грузоперевозчик» и классы «Самолет», «Поезд», «Автомобиль», содержащие объект предметной области в качестве свойства.
3. Создать класс предметной области «Учащийся» и классы «Школьник», «Учащийся ГПТУ» и «Студент», содержащие объект предметной области в качестве свойства.
4. Создать класс предметной области «Музыкальный инструмент» и классы «Ударный», «Струнный», «Духовой», содержащие объект предметной области в качестве свойства.
5. Создать класс предметной области «Работник фирмы» и классы «Менеджер», «Администратор», «Программист», содержащие объект предметной области в качестве свойства.

6. Создать класс предметной области «Домашнее животное» и классы «Собака», «Кошка», «Попугай», содержащие объект предметной области в качестве свойства.
7. Создать класс предметной области «Садовое дерево» и классы «Яблоня», «Вишня», «Груша», содержащие объект предметной области в качестве свойства.
8. Создать класс предметной области «Единица хранения в библиотеке» и классы «Художественная книга», «Научная книга», «Журнал», содержащие объект предметной области в качестве свойства.
9. Создать класс предметной области «Автомобиль» и классы «Легковой автомобиль», «Грузовой автомобиль», «Вездеход», содержащие объект предметной области в качестве свойства.
10. Создать класс предметной области «Плавсредство» и классы «Корабль», «Баржа», «Яхта», содержащие объект предметной области в качестве свойства.
11. Создать класс предметной области «Электростанция» и классы «Теплоэлектростанция», «Гидроэлектростанция», «Атомная станция», содержащие объект предметной области в качестве свойства.
12. Создать класс предметной области «Городской транспорт» и классы «Троллейбус», «Автобус», «Трамвай», содержащие объект предметной области в качестве свойства.
13. Создать класс предметной области «Коммунальная техника» и классы «Уборочная техника», «Мусоровоз», «Автомобиль-разбрасыватель реагентов», содержащие объект предметной области в качестве свойства.
14. Создать класс предметной области «Торговая сеть» и классы «Гипермаркет», «Специализированные магазин», «Рынок», содержащие объект предметной области в качестве свойства.
15. Создать класс предметной области «Школа» и классы «Учитель», «Школьник», «Повар», содержащие объект предметной области в качестве свойства.
16. Создать класс предметной области «Рабочий» и классы «Слесарь», «Электрик», «Монтажник-высотник», содержащие объект предметной области в качестве свойства.

5.2. Классы с одним параметром. Имплементация параметризованного интерфейса. Переопределение методов интерфейса `Iterator` для массива

Задание базируется на уже созданном проекте в рамках предыдущей темы «5.1. Параметризованные методы. Использование классов типа `record`». Задание состоит из ***двух*** *частей*.

Замечание.

Работоспособность первоначального проекта (часть 1), как и результаты внесения в него изменений (часть 2), демонстрируются студентом одновременно при сдаче задания.

5.2.1. Создание параметризованного класса `ArrayWrapper<T>`, имплементирующего интерфейс `Iterator`

Обобщенная формулировка задания части 1. В многофайловый проект, использующий пакеты и ***уже разработанный*** в рамках ***предыдущего*** задания (см. пункт «5.1. Параметризованные методы. Использование классов типа `record`») необходимо внести следующие изменения:

- вместо ***не*** параметризованного класса `Library` необходимо создать ***параметризованный*** `ArrayWrapper<T>` имплементирующий интерфейс `Iterator<T>`, внеся изменения в заголовок файла;
- добавить в качестве свойств параметризованный массив и целочисленное поле, предназначенное для хранения индекса;
- добавить набор методов:
    - `hasNext()`,
    - `next()`,
    - `resetIterator()`,
    - `toString()`;

Пример задания части 1. С использованием предметной области «Основная информация» (`BaseInform`) создать классы, на основе отношения агрегации «Ребенок» (`Child`), «Взрослый» (`Adult`) и «Пенсионер» (`Pensioner`). Вывести на экран содержимое всего массива, а

также сделать выборку по типу записей в массиве. Структура проекта приведена на рисунке ниже (Рисунок 22).

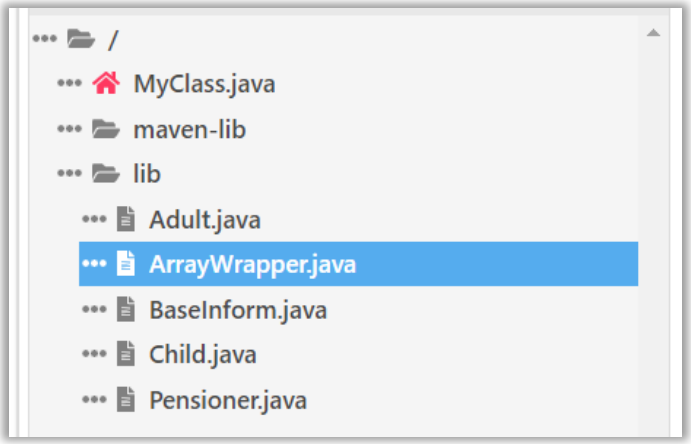


Рисунок 22 – Структура проекта

5.2.1.1. Пример выполнения задания

```
//Без изменений остаются следующие файлы пакета lib
//предыдущего проекта темы 5.1:
//     Adult.java
//     BaseInform.java
//     Child.java
//     Pensioner.java

//Файл ArrayWrapper.java пакета lib содержащим
//параметризованный класс, имплементирующий
//параметризованный интерфейс Iterator
```

/lib/ArrayWrapper.java

```
1  package lib;
2  import java.util.Iterator;
3
4  public class ArrayWrapper<T> implements Iterator<T> {
5      private T[] a;
6      private int index = 0;
7      //конструкторы
8      public ArrayWrapper() {};
9      public ArrayWrapper(T[] a) {
10         this.a = a.clone();
11     }
12     //методы
13     public T[] getData() {
14         return a;
15     }
```

```
    public void setData(T s, int i) {
        a[i] = s;
    }
    @Override
    public boolean hasNext() {
        return this.index < a.length;
    }
    @Override
    public T next() {
        if(!hasNext()) return null;
        return a[this.index++];
    }
    private void resetIterator() {index = 0;}
    public String choise(String strType) {
        String result = "";
        this.resetIterator();
        while(this.hasNext()) {
            var current = this.next();
            if (current.getClass()
                       .getSimpleName()
                       .toUpperCase()
                       .equals(strType.toUpperCase())) {
                result = result + current.toString();
            }
        }
        return result;
    }
    @Override
    public String toString() {
        this.resetIterator();
        String str = this.next().toString();
        while(this.hasNext()) {
             str = str + this.next().toString();
        }
        return str;
    }
}
```

//точка входа в проект

/MyClass.java

```
import lib.*;

public class MyClass
{
    public static void main(String[] arg) {
        Object[] a = {
```

```
                    new Child(new BaseInform("Sidorov", 'm', 15),
                            80),
                    new Child(new BaseInform("Ivanov", 'm', 5),
                            120),
                    new Adult(new BaseInform("Petrov", 'm', 45),
                            1000000),
                    new Pensioner(new BaseInform("Болдырева",
                                                 'f', 60),
                            500)
            };
            ArrayWrapper<Object> obj = new ArrayWrapper<Object>(a);
            System.out.println(obj.toString());
            System.out.println(obj.choise("Pensioner"));
        }
    }
```

Результат работы программы:

```
Child: Sidorov IQ: 80
Child: Ivanov IQ: 120
Adult: Petrov income: 1000000.0
Pensioner: Болдырева pension: 500.0

Pensioner: Болдырева pension: 500.0
```

5.2.1.2. Индивидуальные задания

См. пункт «5.1.2. Индивидуальные задания»

5.2.2. Замена параметризованных типов в классах на супертип `Object`.

Обобщенная формулировка задания части 2. для выполнения и демонстрации работоспособности второй части задания следует:

- создать копию проекта, разработанного в рамках выполнения части 1;
- в созданной копии полностью удалить параметризацию в заголовке класса `ArrayWrapper`, в качестве параметра в имплементируемый параметризованный интерфейс `Iterator` передать класс `Object`;
- в методах класса `ArrayWrapper` заменить параметризованные типы на супертип `Object`;

- разработать ***дополнительный*** метод в соответствие с ***индивидуальным*** заданием ***второй*** части.

Замечание.

Если убрать параметризацию методов файле `ArrayWrapper.java` *и заменить в параметрах типа* `T` *на* `Object`*, то работоспособность программы не измениться, т.к. все классы в Java являются наследниками* `Object`*-а, но использование суперкласса в качестве параметра — это* ***полиморфизм****, а не параметризация.*

5.2.2.1. Пример выполнения задания

```
//пакет lib
```

/lib/ArrayWrapper.java

```
1  package lib;
2  import java.util.Iterator;
3
4  public class ArrayWrapper implements Iterator<Object> {
5      private Object[] a;
6      private int index = 0;
7      //конструкторы
8      public ArrayWrapper() {};
9      public ArrayWrapper(Object[] a) {
10         this.a = a.clone();
11     }
12     //методы
13     public Object[] getData() {
14         return a;
15     }
16     public void setData(Object s, int i) {
17         a[i] = s;
18     }
19     @Override
20     public boolean hasNext() {
21         return this.index < a.length;
22     }
23     @Override
24     public Object next() {
25         if(!hasNext()) return null;
26         return a[this.index++];
27     }
28     private void resetIterator() {index = 0;}
29     public String choise(String strType) {
30         String result = "";
```

```
        this.resetIterator();
        while(this.hasNext()) {
            var current = this.next();
            if (current.getClass()
                    .getSimpleName()
                    .toUpperCase()
                    .equals(strType.toUpperCase())) {
                result = result + current.toString();
            }
        }
        return result;
    }
    @Override
    public String toString() {
        this.resetIterator();
        String str = this.next().toString();
        while(this.hasNext()) {
             str = str + this.next().toString();
        }
        return str;
    }
}
```

//точка входа в проект

/MyClass.java

```
import lib.*;

public class MyClass
{
    public static void main(String[] arg) {
        Object[] a = {
                    new Child(new BaseInform("Sidorov", 'm', 15),
                            80),
                    new Child(new BaseInform("Ivanov", 'm', 5),
                            120),
                    new Adult(new BaseInform("Petrov", 'm', 45),
                            1000000),
                    new Pensioner(new BaseInform("Болдырева",
                                                 'f', 60),
                            500)
        };
        ArrayWrapper obj = new ArrayWrapper(a);
        System.out.println(obj.toString());
        System.out.println(obj.choise("Pensioner"));
    }
}
```

Результат работы программы тот же что и раньше.

5.2.2.2. Индивидуальные задания

Замечание.

Поскольку record-ы являются вариантом реализации иммутабельных классов, то изменить значение одного поля объекта можно только с помощью создания нового объекта вызвав конструктор, в который будут скопированы почти все свойства найденного по заданному признаку объекта, кроме одного или нескольких свойств (в соответствии с заданием), которые надо изменить. При этом изменяемым полям при упомянутом вызове конструктора должны быть присвоены новые значения.

Дополнительный метод для класса `ArrayWrapper`. Написать метод, находящим в списке элемент с ***совпадающим со введенным с клавиатуры*** значением:

1. ***Первого*** поля класса ***предметной области*** и ***изменить*** это значение на новое, введенное с клавиатуры.
2. ***Второго*** поля класса ***предметной области*** и ***изменить*** это значение на новое, введенное с клавиатуры.
3. ***Третьего*** поля класса ***предметной области*** и ***изменить*** это значение на новое, введенное с клавиатуры.
4. ***Первого*** поля ***одного из уточняющих*** классов и ***изменить*** это значение на новое, введенное с клавиатуры.
5. ***Второго*** поля ***одного из уточняющих*** классов и ***изменить*** это значение на новое, введенное с клавиатуры.
6. ***Третьего*** поля ***одного из уточняющих*** классов и ***изменить*** это значение на новое, введенное с клавиатуры.
7. ***Первого*** поля класса ***предметной области*** и изменить ***значение второго*** поля этого элемента на новое, введенное с клавиатуры.
8. ***Первого*** поля класса ***предметной области*** и изменить ***значение третьего*** поля этого элемента на новое, введенное с клавиатуры.
9. ***Второго*** поля класса ***предметной области*** и изменить ***значение третьего*** поля этого элемента на новое, введенное с клавиатуры.
10. ***Третьего*** поля класса ***предметной области*** и изменить ***значение первого*** поля этого элемента на новое, введенное с клавиатуры.
11. ***Третьего*** поля класса ***предметной области*** и изменить ***значение второго*** поля этого элемента на новое, введенное с клавиатуры.
12. ***Первого*** поля класса ***предметной области*** и изменить ***значение первого*** поля одного из ***уточняющих классов*** на новое, введенное с клавиатуры.

13. ***Первого*** поля класса ***предметной области*** и изменить ***значение второго*** поля одного из ***уточняющих классов*** на новое, введенное с клавиатуры.
14. ***Первого*** поля класса ***предметной области*** и изменить ***значение третьего*** поля одного из ***уточняющих классов*** на новое, введенное с клавиатуры.
15. ***Второго*** поля класса ***предметной области*** и изменить ***значение первого*** поля одного из ***уточняющих классов*** на новое, введенное с клавиатуры.
16. ***Второго*** поля класса ***предметной области*** и изменить ***значение третьего*** поля одного из ***уточняющих классов*** на новое, введенное с клавиатуры.

5.3. Двухпараметрические классы. Наследование

Тема посвящена созданию многофайлового проекта с использованием ***нескольких*** пакетов. Как и ранее в этой теме файл `MyClass.java` с классом `MyClass` будет содержать метод `main()` (точку входа в проект).

Обобщенная формулировка задания. В соответствии с заданием следует:

- в пакете `lib.keys` создать три класса-***ключа*** (типа `record`) в соответствии с индивидуальным заданием;
- в пакете `lib.values` создать три класса-***значения*** (типа `record`) в соответствии с индивидуальным заданием;
- каждый из перечисленных классов должен иметь минимум по три свойства;
- в пакете `lib` должны быть созданы параметризованные классы `Node` и `List`;
- в параметризованном классе `Node` должна создаваться соответствие (отображение) ключ-значение;
- параметризованный класс `List` наследует параметризованный класс `Node` и должен иметь набор свойств и методов для связывания двусвязного списка и вывода значений списка в ***обратном*** порядке на экран;
- каждый метод должен иметь перегруженный метод `toString()`;
- в `main()` необходимо создать список минимум из шести пар «ключ-значение» и вывести весь список в обратном порядке на экран, используя методы класса `List`.

Пример задания. Создать список пар «ключ-значение» (Рисунок 23):

- классы-ключи (пакет `lib.keys`):

    - Международный стандартный номер книги (`ISBN`);
    - Международный стандартный номер серийного издания (`ISSN`);
- классы значения (пакет `lib.values`):
    - тип книги (`TypeBook`);
    - тип журнала (`TypeJournal`);
    - газета (`NewsPaper`).
- класс `Node` (пакет `lib`), создающий пары для узла двусвязного списка;
- класс `List` (пакет `lib`), создающий двусвязный список и и методы работы со списком в смысле обратного итератора.

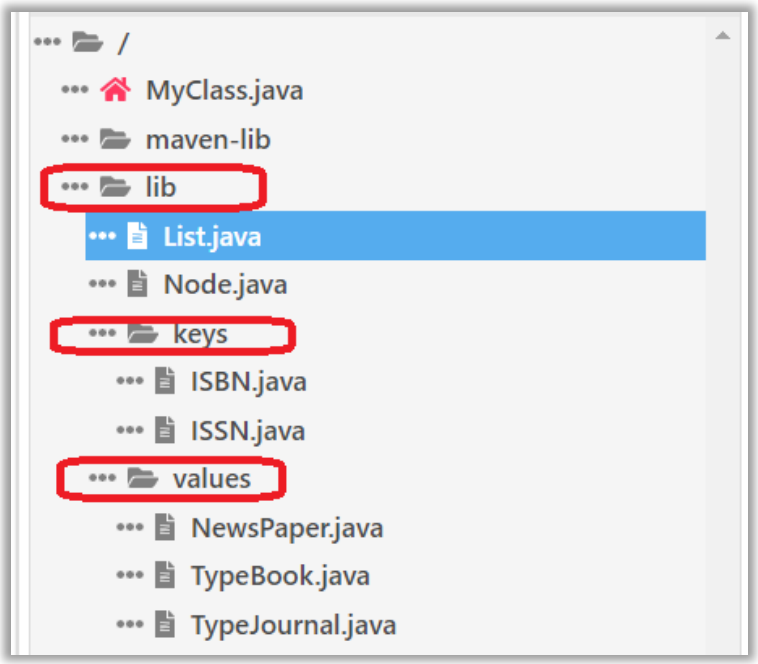


Рисунок 23 – Структура проекта в соответствии с решаемым примером (красным выделены используемые пакеты)

5.3.1. Пример выполнения задания

```
//пакет lib
 /lib/List.java
```

```
1  package lib;
2  import lib.*;
3
4  public class List<T, S> extends Node<T, S> {
```

```
5       //хранение адреса первого элемента списка
6       private Node begin = null;
7       //методы
8       public List() {
9           super();
10      }
11      public void addListEnd(Node node) {
12          if (begin == null) {
13              begin = node;
14              return;
15          }
16          Node current = begin;
17          while (current.next != null) {
18              current = current.next;
19          }
20          current.next =
21              new Node(node.obj1, node.obj2);
22          current.next.prev = current;
23      }
24      public String reversToString() {
25          Node current = begin;
26          while(current.next != null) {
27              current = current.next;
28          }
29          String str = "";
30          while(current != null) {
31              str = str + current.toString();
32              current = current.prev;
33          }
34          return str;
35      }
36  }
```

/lib/Node.java

```
1   package lib;
2
3   public class Node<T, S> {
4       protected T obj1;
5       protected S obj2;
6       //для двусвязного списка
7       protected Node next = null;
```

```
    protected Node prev = null;
    //методы
    public Node() {}
    public Node(T obj1, S obj2) {
        this.obj1 = obj1;
        this.obj2 = obj2;
    }
    @Override
    public String toString() {
        return obj1.toString() + " " +
                obj2.toString() + "\n";
    }
}
```

//пакет keys

/lib/keys/ISBN.java

```
package lib.keys;

public record ISBN(String code) {
    public String toString() {
        return "ISBN " + code;
    }
}
```

/lib/keys/ISSN.java

```
package lib.keys;

public record ISSN(String code) {
    public String toString() {
        return "ISSN " + code;
    }
}
```

//пакет values

/lib/values/Newspaper.java

```
package lib.values;

public record Newspaper(String name) {
    @Override
    public String toString() {
        return name;
    }
}
```

/lib/values/TypeBook.java

```java
package lib.values;

public record TypeBook(String name) {
    public String toString() {
        return name;
    }
}
```

/lib/values/TypeJournal.java

```java
package lib.values;

public record TypeJournal(String name) {
    public String toString() {
        return name;
    }
}
```

//точка входа в проект
/MyClass.java

```java
import lib.*;
import lib.keys.*;
import lib.values.*;

public class MyClass
{
    public static void main(String[] args) {
        List list = new List();
        list.addListEnd(new Node(new ISBN("1111-2222-3333-4444"),
                        new TypeBook("fantasy")));
        list.addListEnd(new Node(new ISSN("0000-9999"),
                        new TypeJournal("sicense")));
        list.addListEnd(new Node(new ISSN("1001-5005"),
                        new Newspaper("New York Times")));
        System.out.println(list.reversToString());
    }
}
```

Результат работы программы

```
ISSN 1001-5005 New York Times
ISSN 0000-9999 sicense
ISBN 1111-2222-3333-4444 fantasy

Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Замечание.

Зачастую работа с дженериками приводит к предупреждениям (выделено красным), во многих случаях это предупреждение можно снять, фактически «уничтожив» параметризацию явной передачей в качестве значений параметров типа `Object`*. Однако в рассматриваемом примере даже это не помогает.*

5.3.2. Индивидуальные задания

1. Создать:
   - классы-ключи: «Автомобиль», «Велосипед», «Повозка»;
   - классы-значения: «Владелец», «Тип», «Регистрационные данные»;
   - сформировать двусвязный список «ключ-значение».
2. Создать:
   - классы-ключи: «Самолет», «Поезд», «Автомобиль»;
   - классы-значения: «Владелец», «Тип», «Регистрационные данные»;
   - сформировать двусвязный список «ключ-значение».
3. Создать:
   - классы-ключи: «Школьник», «Учащийся ГПТУ» и «Студент»;
   - классы-значения: «ФИО», «Характеристика», «Объективные данные»;
   - сформировать двусвязный список «ключ-значение».
4. Создать:
   - классы-ключи: «Ударный муз. инструмент», «Струнный муз. инструмент», «Духовой муз. инструмент»;
   - классы-значения: «Характристики», «Фирма-изготовитель», «ФИО сосбтвенника»;
   - сформировать двусвязный список «ключ-значение».
5. Создать:
   - классы-ключи: «Менеджер», «Администратор», «Программист»;
   - классы-значения: «ФИО», «Паспортные данные», «Образование»;
   - сформировать двусвязный список «ключ-значение».
6. Создать:
   - классы-ключи: «Собака», «Кошка», «Попугай»;
   - классы-значения: «ФИО владельца», «Регистрационные данные», «Сведения о породе»;

- сформировать двусвязный список «ключ-значение».

7. Создать:
   - классы-ключи: «Яблоня», «Вишня», «Груша»;
   - классы-значения: «Название и характеристика сорта», «ФИО создателя», «Сведения об условиях произрастания»;
   - сформировать двусвязный список «ключ-значение».
8. Создать:
   - классы-ключи: «ФИО Автора», «Объективные данные» (год публикации, ISBN или ISSN, количество страниц), «Сведения об издателе»;
   - классы-значения: «Художественная книга», «Научная книга», «Журнал»;
   - сформировать двусвязный список «ключ-значение».
9. Создать:
   - классы-ключи: «Легковой автомобиль», «Грузовой автомобиль», «Вездеход»;
   - классы-значения: «Владелец», «Тип», «Регистрационные данные»;
   - сформировать двусвязный список «ключ-значение».
10. Создать:
    - классы-ключи: «Корабль», «Баржа», «Яхта»;
    - классы-значения: «Владелец», «Тип», «Регистрационные данные»;
    - сформировать двусвязный список «ключ-значение».
11. Создать:
    - классы-ключи (поля геолокация, выдаваемая мощность, срок эксплуатации): «Теплоэлектростанция», «Гидроэлектростанция», «Атомная станция»;
    - классы-значения: «Тип проекта», «ФИО Генерального конструктора», «Характеристика экологической безопасности»;
    - сформировать двусвязный список «ключ-значение».
12. Создать:
    - классы-ключи: «Троллейбус», «Автобус», «Трамвай»;
    - классы-значения: «Тип проекта», «Фирма изготовитель», «Модель подвижного состава»;
    - сформировать двусвязный список «ключ-значение».
13. Создать:
    - классы-ключи: «Уборочная техника», «Мусоровоз», «Автомобиль-разбрасыватель реагентов»;
    - классы-значения: «Фирма изготовитель», «Тип», «Регистрационные данные»;
    - сформировать двусвязный список «ключ-значение».

14. Создать:
    - классы-ключи: «Гипермаркет», «Специализированные магазин», «Рынок»;
    - классы-значения: «Фирма-автор проекта», «Фирма-подрядчик строительства», «Владелец»;
    - сформировать двусвязный список «ключ-значение».
15. Создать:
    - классы-ключи: «Учитель», «Школьник», «Повар»;
    - классы-значения: «ФИО», «Паспортные данные», «Адрес проживания»;
    - сформировать двусвязный список «ключ-значение».
16. Создать:
    - классы-ключи: классы «Слесарь», «Электрик», «Монтажник-высотник»;
    - классы-значения: «ФИО», «Данные о ПТУ», «Данные о техникуме»;
    - сформировать двусвязный список «ключ-значение».

Тема 6. Лямбда-выражения

При выполнении заданий по ***данной*** теме на `JDoodle` (`Advanced IDE`: https://www.jdoodle.com/online-java-compiler-ide/, Рисунок 19) опция `Interactive` должна быть включена (Рисунок 24).

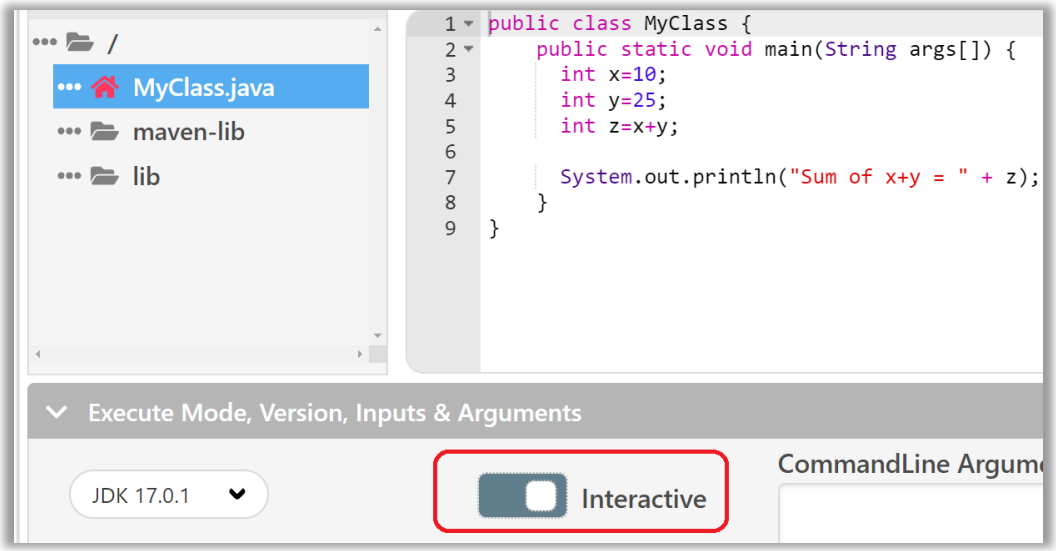


Рисунок 24 – Включенная опция Interactive на `JDoodle Advanced IDE`

6.1. Итерационное решение нелинейных уравнений

Обобщенная формулировка задания. Решить нелинейное уравнение вида $f(x)=0$ с параметром с заданной точностью ε, используя предлагаемые итерационные методы. Предварительно провести графическое отделение одного из корней, т.е. найти отрезок, на котором существует единственный корень. Требуемую точность и начальное приближение ввести с клавиатуры.

Требования:

- реализовать задание в виде многофайлового проекта с одним пакетом `lib`;
- значения начального приближения и точности вести с клавиатуры;
- ввод значений осуществляется с помощью строковых переменных с последующим парсингом (преобразованием) строки в число;
- оформить проект с использованием:
  - лямбда-выражения для функции $f(x)$ из индивидуального задания;
  - в случае реализации метода Ньютона также следует использовать лямбда-выражение для производной функции $f'(x)$.

Итерационные алгоритмы решения нелинейного уравнения. Использовать один из следующих итерационных алгоритмов нахождения решения на отрезке $[a,b]$:

1. Метод ***деления отрезка пополам*** (метод дихотомии).
   Краткое описание метода для отрезка $[a,b]$, содержащего единственный корень. Пусть для определенности $f(a)<0,\ f(b)>0$. В качестве начального приближения корня $\tilde{x}$ принимается середина этого отрезка, т.е. $x_0=(a+b)/2$. Далее исследуем значение функции $f(x)$ на концах отрезков $[a\,;x_0\,]$ и $[\,x_0;b\,]$. Тот из них, на концах которого $f(x)$ принимает значения разных знаков, содержит искомый корень. Поэтому его принимаем в качестве нового отрезка, а вторую половину исходного отрезка $[a;\ b]$ отбрасываем. В качестве первой итерации нахождения корня принимаем середину нового отрезка и т. д. Если длина полученного отрезка становится меньше наперед заданной погрешности, т.е. $|\,b-a\,|<\varepsilon$, счет прекращается.

2. Метод ***простых итераций***:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{M}, \text{где } M > \max_{x\in[a,b]} |f'(x)|, n = \overline{0,\infty}, x_0 = a$$

3. Метод ***Ньютона:***

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\ldots,\ x_0 \in [a,b]$$

4. ***Модифицированный*** метод ***Ньютона:***

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}, n = 0,1,\ldots,\ x_0 \in [a,b]$$

5. Метод ***секущих:***

$$x_{n+1} = x_n - \frac{(x_n - x_{n-1}) f(x_n)}{f(x_n) - f(x_{n-1})}, n = 0,1,\ldots x_0, x_1 \in [a,b]$$

6.1.1. Пример выполнения задания

Решить методом Ньютона с точностью ε нелинейное уравнение $3x - \cos x - 1 = 0$, предварительно отделив один корень.

Многофайловый проект должен иметь структуру, приведенную на рисунке (Рисунок 25).

Рисунок 25 – Структура многофайлового проекта при решении нелинейного уравнения методом Ньютона

//Пакет lib

/lib/Derivative.java

```
1  package lib;
2
3  public interface Derivative {
4      double lambda(double x);
5  }
```

/lib/Function.java

```
1  package lib;
2
3  public interface Function {
4      double lambda(double x);
5  }
```

/lib/Library.java

```
1  package lib;
2
3  public class Library {
4      public static double newtonMethod(double x0,
5                                        double epsilon,
6                                        Function f,
7                                        Derivative df) {
8          double prev = x0;
9          double next = prev -
10                 f.lambda(prev)/df.lambda(prev);
11         while (Math.abs(prev-next) >= epsilon) {
12             prev = next;
13             next = prev -
14                 f.lambda(prev)/df.lambda(prev);
15         }
16         return prev;
17     }
18 }
```

//точка входа в проект
/MyClass.java

```
1  import java.util.Scanner;
2  import lib.*;
3
4  public class MyClass
5  {
6      public static void main(String[] args) {
7          String exseptReport = "Неверный формат " +
8                                "ввода числа";
9          Scanner in = new Scanner(System.in);
10         double x0, epsilon;
11
12         while(true) {
13             System.out.println("Введите начальное " +
14             "приближ. x0 и точность epsilon: ");
15         try {
16             x0 = Double.parseDouble(in.nextLine());
17             epsilon = Double.parseDouble(in.nextLine());
18         }
19         catch(NumberFormatException e) {
20             System.out.println(exseptReport);
21             return;
22         }
23         if (epsilon > 0 && epsilon < 0.05) break;
24             System.out.println("Точность не верна." +
25             "Попробуйте еще раз!!!");
26         }
27         lib.Function f = (x) -> 3 * x - Math.cos(x) - 1;
28         lib.Derivative df = (x) -> 3 + Math.sin(x);
29         System.out.println("Корень равен = " +
30                 lib.Library
31                    .newtonMethod(x0, epsilon, f, df));
32     }
33 }
```

Замечание.

Ввод значений осуществляется по одному и после каждого значения необходимо нажать клавишу `Enter`*. Это делается для того, чтобы гарантировать, что каждое число было представлено в виде отдельной строки.*

Результат работы программы:

```
Введите начальное приближ. x0 и точность epsilon:
1 0.005
Неверный формат ввода числа
```

6.1.2. Индивидуальные задания

Замечание.

При решении нелинейного уравнения методом деления отрезка пополам файл `Derivative.java` *не нужен.*

Требования к выполнению индивидуальных заданий:

- выполнить определение каждого корня для переменного параметра α, изменяющемся в отдельном ***статическом*** методе на заданном интервале;
- реализовать по два метода указанных в таблице (Таблица 1).

Таблица 1 – Таблица функций и номеров итерационных методов

№	$f(x)$	Интервал для α	Шаг изменения α	Итерационный метод
1	$x^2\cos 2x+1+\alpha$	1:2	0.1	(1), (3)
2	$2x-\alpha\cos x$	0:1	0.1	(2), (4)
3	$(x-1+\alpha)^3+0.5e^x$	0:0.5	0.1	(1), (5)
4	$\alpha x-\cos x-1$	3:4	0.1	(2), (3)
5	$x^4+\alpha x-1$	1:5	1	(1), (4)
6	$x^4+\cos x-\alpha$	3:7	0.5	(2), (5)
7	$3x-e^{\alpha x}$	1:5	1	(1), (3)
8	$e^x-1-\alpha x^3$	1:5	1	(2), (4)
9	$\alpha x-\ln x-5$	0.5:4	0.5	(1), (5)
10	$e^{x+\alpha}-x+1$	1:5	1	(2), (3)
11	$e^{1/x}-(x+1)^2+\alpha$	0:5	1	(1), (4)
12	$\alpha x^3-\sin x$	0:1	0.1	(2), (5)
13	$e^x-6x-\alpha$	3:10	1	(1), (3)
14	$x^2+\cos x-\alpha$	3:7	0.5	(2), (4)
15	$x-\sin x-\alpha$	2:3	0.1	(1), (5)
16	$\alpha x+e^x$	1:5	1	(2), (3)

6.2. Приближенное вычисление определенных интегралов

Обобщенная формулировка задания. Найти приближенное значение определенного интеграла $\int_a^b f(x)dx$ с заданной точностью ε, используя две из предложенных квадратурных формул.

Для достижения заданной точности использовать двойной пересчет: вычислить интеграл вначале для n узлов, затем – для $2 \cdot n$; сравнить полученные результаты: если $|I_n - I_{2n}| < \varepsilon$, то $I \approx I_{2n}$, в противоположном случае вычислить интеграл для $4 \cdot n$ и т.д.

Подынтегральную функцию $f(x)$ и предлагаемую квадратурную формулу оформить в виде ***лямбда-выражения***. Значения a, b, ε вести с клавиатуры.

Требования:

- реализовать задание в виде многофайлового проекта с тремя пакетами `lib`, `interfaces`, `library`;
- значения a, b, ε вести с клавиатуры;
- оформить проект с использованием:
  - лямбда-выражения для подынтегральной функции $f(x)$ из индивидуального задания;
  - лямбда-выражения для двух квадратурных формул из индивидуального задания.

Формулы для вычисления интегралов. Пусть задано разбиение отрезка интегрирования $[a, b]$ на n интервалов узловыми точками $x_i = a + i \cdot h$ $(i = \overline{0, n})$ с шагом $h = \frac{b-a}{n}$. Тогда можно рассмотреть следующие простейшие формулы для приближенного вычисления определенных интегралов:

1. Составная формула трапеций:
$$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(x_0) + f(x_n)\right) + h \cdot \sum_{i=1}^{n-1} f(x_i),$$
2. Формула левых прямоугольников:
$$\int_a^b f(x)dx \approx h \cdot \sum_{i=0}^{n-1} f(x_i).$$
3. Формула правых прямоугольников:

$$\int_a^b f(x)dx \approx h \cdot \sum_{i=1}^{n} f(x_i).$$

4. Формула средних прямоугольников:

$$\int_a^b f(x)dx \approx h \cdot \sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right).$$

5. Нижняя сумма Дарбу:

$$\int_a^b f(x)\,dx \approx h \cdot \sum_{i=0}^{n-1} min\{f(x_i), f(x_{i+1})\}.$$

6. Верхняя сумма Дарбу:

$$\int_a^b f(x)\,dx \approx h \cdot \sum_{i=0}^{n-1} max\{f(x_i), f(x_{i+1})\}.$$

Пример задания. Используя квадратурную формулу левых прямоугольников (2), вычислить приближенное значение интеграла $\int_a^b \sin x \cdot e^x dx$ с точностью ε. Многофайловый проект должен иметь структуру, приведенную на рисунке ниже (Рисунок 26).

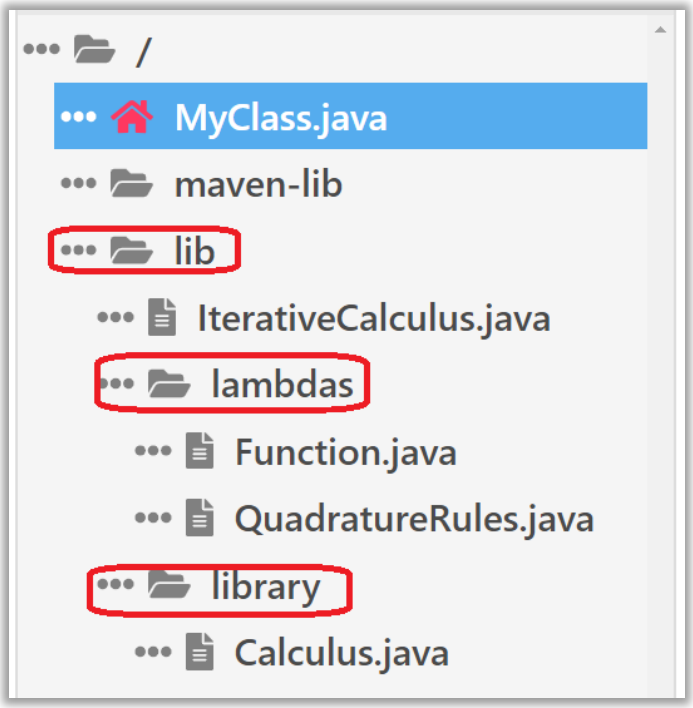


Рисунок 26 – Структура многофайлового проекта при вычислении интеграла с помощью правила левых прямоугольников (красным выделены пакеты)

6.2.1. Пример выполнения задания

//пакет lib

/lib/IterativeCalculus.java

```
1  package lib;
2  import lib.lambdas.*;
3
4  public class IterativeCalculus {
5      public static double sumIterator (double a,
6                                        double b,
7                                        double epsilon,
8                                        QuadratureRules sum,
9                                        Function f) {
10         int n = 10;
11         double current = sum.lambda(a, b, n, f);
12         double next = sum.lambda(a, b, 2 * n, f);
13         do {
14             current = sum.lambda(a, b, n, f);
15             next = sum.lambda(a, b, 2 * n, f);
16             n = 2 * n;
17         }
18         while (Math.abs(current - next) > epsilon);
19         return current;
20     }
21 }
```

//пакет lib.lambdas

/lib/lambdas/Function.java

```
1  package lib.lambdas;
2
3  public interface Function {
4      double lambda(double x);
5  }
```

/lib/lambdas/QuadratureRules.java

```
1  package lib.lambdas;
2
3  public interface QuadratureRules {
4      double lambda(double a, double b, int n, Function f);
5  }
```

//пакет lib.library

/lib/library/Calculus.java

```
1  package lib.library;
2  import lib.lambdas.*;
3
4  public class Calculus {
5
6      public static double LeftRectangles (double a,
7                                           double b,
8                                           int n,
9                                           Function f) {
10         double h = Math.abs(b - a) / n;
11         double result = 0;
12         for(int i = 0; i < n; i++) {
13             result = result + f.lambda(a + i * h);
14         }
15         result = h * result;
16         return result;
17     }
18 }
```

//точка входа в проект

/MyClass.java

```
1  import java.util.Scanner;
2  import lib.*;
3  import lib.lambdas.*;
4  import lib.library.*;
5
6  public class MyClass
7  {
8      public static void main(String[] args) {
9          String exseptReport = "Неверные формат ввода числа";
10         Scanner in = new Scanner(System.in);
11         double a = 0, b = 1, epsilon = 1;
12
13         while(true) {
14             System.out.println("Введите интервал интегрирования " +
15                                "[a, b] и точность epsilon: ");
16             try {
17                 a = Double.parseDouble(in.nextLine());
18                 b = Double.parseDouble(in.nextLine());
19                 epsilon = Double.parseDouble(in.nextLine());
20             }
21             catch (NumberFormatException e) {
22                 System.out.println(exseptReport);
23             }
```

```
            if ( (epsilon < 0.05) && (epsilon > 0) ) break;
            System.out.println("Точность не верна. " +
                               "Попробуйте еще раз!!!");
        }
        lib.lambdas.Function f = (x) -> Math.sin(x) * Math.exp(x);
        lib.lambdas.QuadratureRules sum = (begin, end, n, funct) ->
                    Calculus.LeftRectangles(begin, end, n, funct);
        System.out.println("Корень равен = " +
                IterativeCalculus.sumIterator(a, b, epsilon, sum, f));
    }
}
```

Результат работы программы:

```
Введите интервал интегрирования [a, b] и точность epsilon:
1
2
0.005
Интеграл равен = 4.494596410233291
```

Замечание.

Ввод значений осуществляется по одному и после каждого значения необходимо нажать клавишу `Enter`*. Это делается для того, чтобы гарантировать, что каждое число было представлено в виде отдельной строки.*

6.2.2. Индивидуальные задания

Реализовать итерационное вычисление интегралов для указанных функций и номеров квадратурных формул (Таблица 2).

Таблица 2 - Варианты подынтегральных функций и квадратурных формул

№	$f(x)$	Квадратурные формулы	Точность
1	2	3	4
1	$\sqrt{x}$	(1), (2)	0.005
2	e^{x^2}	(2), (3)	0.0005
3	$1/(1+x)$	(3), (4)	0.00005
4	$1/(1+x^3)$	(4), (5)	0.005
5	e^{-x^2}	(1), (2)	0.005

1	2	3	4
6	$e^{1/x}$	(2), (3)	0.0005
7	$\sin(x)/x$	(3), (4)	0.005
8	$\cos(x)/(x^3+1)$	(4), (5)	0.00005
9	$\sin x^2$	(1), (2)	0.0005
10	$\cos x^2$	(2), (3)	0.005
11	$1/1-x+x^2$	(3), (4)	0.00005
12	$1/\sqrt{x^4}$	(4), (5)	0.0005
13	$1/x$	(1), (2)	0.005
15	$\ln x$	(2), (3)	0.005
15	$\sqrt{x}\cos x$	(3), (4)	0.00005
16	$\sqrt{x}\sin x$	(4), (5)	0.005
17	$\sin(x)/(1+x^2)$	(1), (2)	0.0005
18	$\cos(x)/x^2$	(2), (3)	0.005

6.3. Лямбда-выражения как результаты методов

Обобщенная формулировка задания. Найти приближенное значение определенного интеграла $\int_a^b f(x)dx$ с заданной точностью ε, используя:

- две из предложенных квадратурных формул;
- две подынтегральные функции в соответствии с индивидуальным заданием.

Для достижения заданной точности использовать двойной пересчет: вычислить интеграл вначале для n разбиений отрезка, затем – для $2n$; сравнить полученные результаты: если $|I_n - I_{2n}| < \varepsilon$, то $I \approx I_{2n}$, в противоположном случае вычислить интеграл для $4n$ и т.д.

Замечание.

Квадратурные формулы в соответствии с их номерами в индивидуальных заданиях этого пункта следует взять из предыдущего пункта (см. 6.2. Приближенное вычисление определенных интегралов).

Требования:

- реализовать задание в виде многофайлового проекта с использованием четырех пакетов:
  - `lib`,
  - `lib.changeLambda`,
  - `lib.lambda`,
  - `lib.library`;
- значения a, b, ε вести с клавиатуры;
- оформить в виде отдельных методов возвращающих лямбда-выражения:
  - выбор подынтегральной функции $f(x)$ из индивидуального задания;
  - выбор одной из квадратурных формул из индивидуального задания.
- название пакета `library` сменить на `rules`.

Пример выполнения задания. Используя две квадратурные формулы левых и правых прямоугольников, вычислить с точностью ε приближенное значение интегралов по выбору:

- $\int_a^b \sin x \cdot e^x dx$;
- $\int_a^b x dx$.

Многофайловый проект должен иметь структуру, приведенную на рисунке (Рисунок 27).

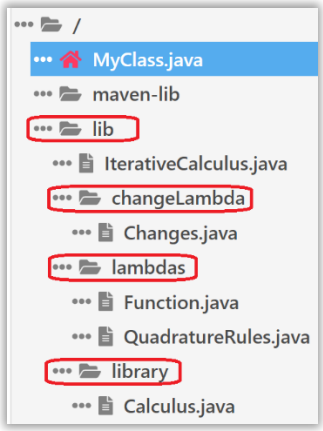


Рисунок 27 – Структура многофайлового проекта при вычислении интеграла с помощью правила левых прямоугольников (красным выделены пакеты)

6.3.1. Пример выполнения задания

```
//пакет lib
//файл IterativeCalculus.java остается таким же как и в примере предыдущего задания (см. 6.2.1. Пример выполнения задания)

//пакет lib.changeLambda
```

/lib/changeLambda/Changes.java

```java
package lib.changeLambda;
import lib.lambdas.*;
import lib.library.*;

public class Changes {
    public static Function selectIntegrandFunct(int number){
        switch(number){
            case 1: return (x) -> Math.sin(x) * Math.exp(x);
            case 2: return (x) -> x;
            default: return (x) -> 0;
        }
    }

    public static QuadratureRules selectQuadrRules(int number){
        switch(number){
            case 1: return (a, b, n, f) -> Calculus
                                               .leftRectangles(a, b, n, f);
            case 2: return (a, b, n, f) -> Calculus
                                               .rightRectangles(a, b, n, f);
            default: return (a, b, n, f) -> 0;
        }
    }
}
```

```
//Пакет lib.lambda
//файлы
//    Function.java,
//    Quadraturerules.java остаются таким же как и в
//примере предыдущего задания
//(см. 6.2.1. Пример выполнения задания

//пакет lib.library
```

/lib/library/Calculus.java

```java
package lib.library;
import lib.lambdas.*;
import lib.*;

public class Calculus {
    //правило левых прямоугольников
    public static double leftRectangles(double a,
                                        double b,
                                        int n, Function f) {
        double h = Math.abs(b - a) / n;
        double result = 0;
        for(int i = 0; i < n; i++) {
            result = result + f.lambda(a + i * h);
        }
        result = h * result;
        return result;
    }
    //правило правых прямоугольников
    public static double rightRectangles(double a,
                                         double b,
                                         int n, Function f) {
        double h = Math.abs(b - a) / n;
        double result = 0;
        for(int i = 1; i <= n; i++) {
            result = result + f.lambda(a + i * h);
        }
        result = h * result;
        return result;
    }
}
```

//точка входа в проект

/MyClass.java

```java
import java.util.Scanner;
import lib.*;
import lib.lambdas.*;
import lib.library.*;
import lib.changeLambda.*;

public class MyClass
{
    public static void main(String[] args) {
        String exseptStr = "Неверные формат ввода числа";
        Scanner in = new Scanner(System.in);
        double a = 0, b = 1, epsilon = 1;

```

```
14          while(true) {
15              System.out.println("Введите интервал интегрирования " +
16                                 "[a, b] и точность epsilon: ");
17              try {
18                  //каждое число вводится в отдельной строке
19                  a = Double.parseDouble(in.nextLine());
20                  b = Double.parseDouble(in.nextLine());
21                  epsilon = Double.parseDouble(in.nextLine());
22              }
23              catch (NumberFormatException e) {
24                  System.out.println(exseptStr);
25              }
26              if ( (epsilon < 0.05) && (epsilon > 0) ) break;
27              System.out.println("Точность не верна. " +
28                                 "Попробуйте еще раз!!!");
29          }
30          System.out.println("Интеграл равен = " +
31                      lib.IterativeCalculus
32                         .sumIterator(a, b, epsilon,
33                                      Changes.selectQuadrRules(2),
34                                      Changes.selectIntegrandFunct(2)));
35      }
36  }
```

6.3.2. Индивидуальные задания

Реализовать итерационное вычисление интегралов для указанных функций и номеров квадратурных формул (Таблица 3).

Таблица 3 - Варианты подынтегральных функций и квадратурных формул

№	Первая подынтегральная $f(x)$	Вторая подынтегральная $f(x)$	Квадратурные формулы	Точность
1	2	3	4	5
1	$x^2 \cos 2x + 1$	$1/(1+x^3)$	(1), (2)	0.005
2	$2x - \cos x$	e^{-x^2}	(2), (3)	0.0005
3	$(x-1)^3 + 0.5 \cdot e^x$	$e^{1/x}$	(3), (4)	0.00005
4	$x - \cos x - 1$	$\sin x / x$	(4), (5)	0.005
5	$x^4 + x - 1$	$\sin x / x$	(1), (2)	0.005
6	$x^4 + \cos x$	$\sin x^2$	(2), (3)	0.0005
7	$3x - e^x$	$\cos x^2$	(3), (4)	0.005
8	$e^x - 1 - x^3$	$1/1 - x + x^2$	(4), (5)	0.00005
9	$x - \ln x - 5$	$1/\sqrt{x^4}$	(1), (2)	0.0005

1	2	3	4	5
10	$e^{x+5}-x+1$	$1/x$	(2), (3)	0.005
11	$e^{1/x}-(x+1)^2$	$\ln x$	(3), (4)	0.00005
12	$x^3-\sin x$	$\sqrt{x}\cos x$	(4), (5)	0.0005
13	e^x-6x	$\sqrt{x}\sin x$	(1), (2)	0.005
15	$x^2+\cos x$	$\sin(x)/(1+x^2)$	(2), (3)	0.005
15	$x-\sin x$	$\cos(x)/x^2$	(3), (4)	0.00005
16	$x+e^x$	$\sqrt{x}$	(4), (5)	0.005

Тема 7. Коллекции, реализующие интерфейс `List`. Работа с текстовыми файлами

Тема посвящена созданию многофайлового проекта и содержит пример работы с коллекцией `ArrayList`, который заполняет коллекцию либо целыми числами, либо числами с плавающей точкой удвоенной точности. Выбор типа хранения для чисел осуществляется типом чисел, задающих диапазон генерации случайных значений:

- если диапазон, вводимый с клавиатуры, является целыми числами, то генерируется коллекция с целыми значениями;
- если диапазон, будет типа `double`, то генерируется коллекция со значениями типа `double`.

Изменение типа диапазона возможно потому, что первоначальный ввод чисел осуществляется, через строку и только после ввода преобразуется в целый или тип с плавающей точкой в зависимости от наличия разделителей типа точка `'.'` или `','`.

Как и ранее при вводе через строку допускается набор произвольного набора строковых символов в составе записи числа.

Обобщенная формулировка задания. Необходимо:

- создать коллекцию типа `ArrayList`;
- заполнить эту коллекцию тем типом чисел, которые определяет введенный с клавиатуры диапазон генерации значений;
- записать созданный одномерный вектор типа `ArrayList` в текстовый файл `"vector.txt"` с помощью методов класса `WorkWithFile` (файл `WorkWithFile.java`, Рисунок 28);
- создать класс `Library` (файл `Library.java`, Рисунок 28), имеющий в качестве свойств:

    - коллекцию типа `ArrayList`;
    - значения параметров (ранее они возвращались методами, теперь результаты работы методов необходимо вернуть через поля класса `Library`), которые необходимо получить после обработки коллекции;
- класс `Library` (файл `Library.java`, Рисунок 28) должен иметь методы:
    - конструктор, получающий в качестве параметра инициализирующую коллекцию;
    - геттеры для свойств метода;
    - метод в соответствии с частью 1 индивидуального задания для обработки одномерной коллекции;
    - метод в соответствии с частью 2 индивидуального задания для обработки одномерной коллекции как двумерного массива;
- прочитать одномерный вектор типа `ArrayList` из текстового файла `"vector.txt"` с помощью методов класса `WorkWithFile` (файл `WorkWithFile.java`, Рисунок 28);
- проинициализировать прочитанным объектом класса `ArrayList` объект класса `Library` с помощью его конструктора;
- вызвать метод класса `Library` для обработки его свойства в виде коллекции согласно индивидуальному заданию из части 1;
- вывести значение, определенное методом из индивидуального задания на экран;
- предполагая, что вектор можно представить в виде квадратного двумерного массива ($\sqrt{n}$ – целое число, где n – количество элементов в одномерном массиве) записать созданную на первом шаге коллекцию в файл в виде матрицы с помощью методов класса `WorkWithFile` (файл `WorkWithFile.java`, Рисунок 28);
- прочитать из многострочного файла в виде двумерного массива значения и преобразовать их обратно в одномерный массив с помощью методов класса `WorkWithFile` (файл `WorkWithFile.java`, Рисунок 28);
- проинициализировать этими значениями объект класса `Library` с помощью его конструктора;
- предполагая, что свойство в виде одномерной коллекции объекта класса `Library` представляет собой квадратный двумерный массив, вызвать метод класса `Library` для его

обработки методом согласно индивидуальному заданию из части 2;

- вывести полученное значение на экран;

В процессе выполнения задания учащемуся понадобятся стандартные для всех заданий методы инициализации коллекции класса `InitCollection` (файл `InitCollection.java`), а также методы конвертации вводимых с помощью строк чисел класса `StringToNumber` (файл `StringToNumber.java`)

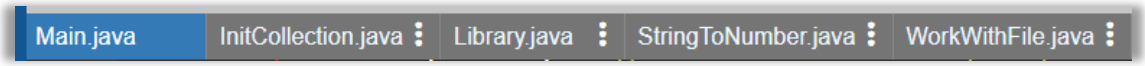


Рисунок 28 – Структура многофайлового проекта с безымянным пакетом «по умолчанию» в соответствии с решаемым примером (красным выделены используемые пакеты)

Пример индивидуального задания. Индивидуальные задания касаются изменений только в классе `Library` (файл `Library.java`). Этот класс должен содержать методы согласно:

- в одномерной коллекции-свойства объекта класса `Library` из `n` чисел найти минимальный ($\sqrt{n}$ – целое число) (часть 1);
- интерпретируя одномерную коллекцию с количеством элементов n как двумерный квадратный массив с размерностью $\sqrt{n} \times \sqrt{n}$ найти максимальный элемент на главной диагонали (часть 2).

7.1. Пример выполнения задания.

```
//Файл InitCollection.java
```

Main.java | InitCollection.java | Library.java | StringToNumber.java | WorkWithFile.java

```
1  import java.util.*;
2
3  public class InitCollection {
4      public static Number initBoundaries() {
5          Locale locale = new Locale.Builder()
6                                    .setLanguage("en")
7                                    .setRegion("US").build();
8          Scanner in = new Scanner(System.in);
9          in.useLocale(locale);
10         String str = in.nextLine();
11         if (str.indexOf(',') >= 0 || str.indexOf('.') >= 0) {
12             str = StringToNumber.doubleDelInvalid(str);
13             return Double.parseDouble(str);
14         }
```

```
        else {
            str = StringToNumber.intDelInvalid(str);
            return Integer.parseInt(str);
        }
    }
    //обработка исключения: типы MAX и MIN не совпадают
    public static <T extends Number>
                boolean isEqualsTypes(final T MIN,
                                      final T MAX) {
        if (!MIN.getClass().getSimpleName().toString()
                .equals(MAX.getClass()
                        .getSimpleName().toString()) )
        try {
            throw new Exception("типы MIN и MAX не совпадают");
        }
        catch(Exception ex) {
            System.out.println(ex.getMessage());
            return false;
        }
        return true;
    }

    //обработка исключения: величина MAX < величины MIN
    public static <T extends Number>
                boolean isCorrectOrder(final T MIN,
                                       final T MAX) {
        boolean correctOrder;
        if (MIN.toString().indexOf(',') >= 0
            || MIN.toString().indexOf('.') >= 0) {
            correctOrder = Double
                           .parseDouble(MIN.toString())
                           < Double
                             .parseDouble(MAX.toString());
        }
        else {
            correctOrder = Integer.parseInt(MIN.toString())
                           < Integer.parseInt(MAX.toString());
        }
        if (!correctOrder)
        try {
            throw new Exception("Величина MAX < величины MIN");
        }
        catch(Exception ex) {
            System.out.println(ex.getMessage());
            return false;
        }
        return correctOrder;
    }

    public static <T extends Number>
                        Double valueRandom(T a, T b) {
        return Math.random()
```

```
                * (b.doubleValue() - a.doubleValue())
                  + a.doubleValue();
    }

    public static <T extends Number>
                    ArrayList generationArray(final T MIN,
                                              final T MAX,
                                              int n) {
      ArrayList array = new ArrayList();
        if(MIN instanceof Double) {
            for(int i = 0; i < n; i++) {
               array.add(valueRandom(MIN, MAX));
            }
        }
        else if (MIN instanceof Integer) {
            for(int i = 0; i < n; i++) {
               array.add(valueRandom(MIN, MAX).intValue());
            }
        }
        return array;
    }
}
```

//Файл Library.java

Main.java | InitCollection.java | **Library.java** | StringToNumber.java | WorkWithFile.java

```
import java.util.*;

public class Library<T extends Number> {
    private ArrayList<T> a;
    private T min;
    private T max;
    //методы класса
    public Library(ArrayList a) {
        this.a = new ArrayList(a);
    }
    public T getMin() {
        return min;
    }
    public T getMax() {
        return max;
    }

    public static <T> boolean moreThan(T a, T b) {
        boolean result;
        if (a.toString().indexOf('.') >= 0) {
            result = Double.parseDouble(a.toString())
                   > Double.parseDouble(b.toString());
        }
        else {
            result = Integer.parseInt(a.toString())
```

```
                    > Integer.parseInt(b.toString());
        }
        return result;
    }

    public void min() {
        min = a.get(0);
        for(int i = 1; i < a.size(); i++) {
            if (Library.moreThan(min, a.get(i))) {
                min = a.get(i);
            }
        }
    }

    public void maxDiagonalMatrix() {
        max = a.get(0);
        int n = (int) Math.sqrt(a.size());
        for(int i = 1; i < n; i++) {
            if (!(Library.moreThan(max, a.get(i * n + i)))) {
                max = a.get(i * n + i);
            }
        }
    }

    public String vectorToString() {
        String result = "";
        int n = a.size();
        for(int i = 0; i < n; i++) {
            result = result + a.get(i) + " ";
        }
        return result;
    }

    public String matrixToString() {
        String result = "";
        int n = (int) Math.sqrt(a.size());
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++) {
                if (a.get(0) instanceof Integer) {
                    result = result
                            + String.format("%4d ",
                                    a.get(i * n + j));
                }
                else {
                    result = result
                       + String.format("%7.3f ",
                                a.get(i * n + j));
                }
            }
            if (i < n - 1) result = result + "\n";
        }
        return result;
    }
}
```

//Файл StringToNumber.java

Main.java | InitCollection.java | Library.java | StringToNumber.java | WorkWithF

```
1  public class StringToNumber {
2      //статический метод подготовка к переводу строки в
3      //целое число
4      static String intDelInvalid(String str) {
5          StringBuffer a = new StringBuffer(str);
6          String validChar = "0123456789";
7          //инициализация индекса символа введенной
8          //строки в зависимости от знака числа
9          int i = (a.charAt(0) == '-'? 1: 0);
10         for(; i < a.length(); ){
11             if(validChar.indexOf(a.charAt(i)) < 0) {
12                 a.delete(i, i + 1);
13             }
14             else i++;
15         }
16         return a.toString();
17     }
18     //экземплярный метод подготовки к переводу
19     //в число double
20     static String doubleDelInvalid(String str) {
21         //предварительная замена всех запятых на точку
22         StringBuffer a =
23                 new StringBuffer(str.replace(',', '.'));
24
25         //поиск первой точки слева и удаление всех точек
26         //справа от нее
27         for(int i = a.toString().lastIndexOf('.');
28                 a.toString().indexOf('.') < i; ) {
29             a.delete(i, i + 1);
30             i = a.toString().lastIndexOf('.');
31         }
32         //строка допустимых символов дополняется точкой
33         String validChar = ".0123456789";
34         //повторение алгоритма удаления недопустимых
35         //символов из предыдущего метода
36         int i = (a.charAt(0) == '-'? 1: 0);
37         for(; i < a.length(); ) {
38             if(validChar.indexOf(a.charAt(i)) < 0) {
39                 a.delete(i, i + 1);
40             }
41             else i++;
42         }
43         return a.toString();
44     }
45 }
```

//Файл WorkWithFile.java

Main.java | InitCollection.java | Library.java | StringToNumber.java | WorkWithFile.java

```
1  import java.util.*;
2  import java.io.*;
3
4  public class WorkWithFile {
5      public static void writeVectorTXT(ArrayList v,
6                                        String nameFile)
7                                        throws IOException {
8          //из строки-реузльтата toString() убираем квадратные
9          //скобки и запятые для записи в файл
10         String textForFile = v.toString()
11                             .replaceAll("^\\[|\\]$", "")
12                             .replaceAll(",","");
13         //создаем поток
14         FileOutputStream fos = new FileOutputStream(nameFile);
15         //конвертрование строки матрицы в массив байтов
16         byte[] buffer = textForFile.getBytes();
17         //запись файла чисел одной строкой
18         fos.write(buffer, 0, buffer.length);
19         //закрываем поток
20         fos.close();
21     }
22
23     public static ArrayList readVectorTXT(String nameFile)
24                                           throws IOException {
25         //создаем входящий поток и инициализируем им
26         //объект Scanner
27         Scanner in = new Scanner(new FileInputStream(nameFile));
28         //читаем файл одной строкой сразу
29         String s = in.nextLine();
30         //уничтожаем объект in, а через него и закрываем поток
31         in.close();
32         //преобразование целой строки в массив строк (в качестве
33         //разделителя - пробел)
34         String[] str = s.split(" ");
35         ArrayList v = new ArrayList();
36         //преобразование массива строк в число и присвоение их
37         //значения коллекции v
38         if (str[0].indexOf(".") >= 0) {
39             for(int i = 0; i < str.length; i++) {
40                 v.add(Double.parseDouble(str[i]));
41             }
42         }
43         else {
44             for(int i = 0; i < str.length; i++) {
45                 v.add(Integer.parseInt(str[i]));
46             }
47         }
48         return v;
49     }
50
```

```
51      public static void writeMatrixTXT(ArrayList v,
52                                        String nameFile)
53                                        throws IOException {
54          int n = (int) Math.sqrt(v.size());
55          //открытие потока вывода
56          FileOutputStream fos = new FileOutputStream(nameFile);
57          //построчнный вывод матрицы в файл nameFile
58          for(int i = 0; i < n; i++) {
59              String strMatrixForFile = "";
60              for(int j = 0; j < n; j++) {
61                  strMatrixForFile = strMatrixForFile
62                                     + v.get(i * n + j) + " ";
63              }
64              //для указания конца строки матрицы должен
65              //стоять символ "\n"
66              if (i < n - 1) {
67                  strMatrixForFile = strMatrixForFile + "\n";
68              }
69              //конвертрование строки матрицы в массив байтов
70              byte[] buffer = strMatrixForFile.getBytes();
71              //построчная запись матрицы
72              fos.write(buffer, 0, buffer.length);
73          }
74          //закрытие потока вывода
75          fos.close();
76      }
77
78      public static int numberStringInTXTFile(String nameFile)
79                                              throws IOException {
80          Scanner in = new Scanner(new FileInputStream(nameFile));
81          int n = 0;
82          //подсчет количества строк в текстовом файле
83          try {
84              //читаем построчно файл методом nextLine() пока
85              //не возникает исключение NoSuchElementException
86              while (in.nextLine().getClass()
87                      .getSimpleName().equals("String")) n++;
88          }
89          catch (NoSuchElementException ex) {
90              in.close();
91           }
92          finally {
93              return n;
94          }
95      }
96
97      public static ArrayList readMatrixTXT(String nameFile)
98                                            throws IOException {
99          //построчное чтение из файла
100         Scanner in = new Scanner(new FileInputStream(nameFile));
101         ArrayList v = new ArrayList();
102         String s;
103         while(in.hasNextLine()){
104             s = in.nextLine();
```

```
            String words[] = s.split(" ");
            if (words[0].indexOf(".") >= 0) {
                for(int i = 0; i < words.length; i++) {
                    v.add(Double.parseDouble(words[i]));
                }
            }
            else {
                for(int i = 0; i < words.length; i++) {
                    v.add(Integer.parseInt(words[i]));
                }
            }
        }
        in.close();
        return v;
    }
}
```

//Файл Main.java

Main.java | InitCollection.java | Library.java | StringToNumber.java | WorkWithFile.java | vecto

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main (String args[]) {
        //количество элементов в одномерном массиве
        final int N = 16;
        //вспомогательные сообщения
        String callMin = "Введите значение НИЖНЕЙ "
                       + "границы генерации чисел в коллекции:";
        String callMax = "Введите значение ВЕРХНЕЙ "
                       + "границы генерации чисел в коллекции:";
        String reportPart1 = "\n--ОДНОМЕРНЫЙ МАССИВ\n";
        String reportPart2 = "\n--ИНТЕРПРЕТАЦИЯ ВЕКТОРА "
                           + "КАК МАТРИЦЫ\n";
        //начало алгоритма
        System.out.println(callMin);
        //нижняя граница генерации
        var min = InitCollection.initBoundaries();
        System.out.println(callMax);
        //верхняя граница генерации
        var max = InitCollection.initBoundaries();
        //проверка корректности
        if (!(InitCollection.isEqualsTypes(min, max)
             && InitCollection.isCorrectOrder(min, max))) return;
        try {
            //генерация значений и запись одномерной коллекции
            //чисел в файл (количество чисел равно N)
            WorkWithFile.writeVectorTXT(InitCollection
                                        .generationArray(min, max, N
                                        "vector.txt");
            //Чтение сохраненного массива чисел из файла и
            //инициализация им коллекции
            ArrayList vector = WorkWithFile
                             .readVectorTXT("vector.txt");
```

```
35              //инициализация объекта класса прочитанным вектором и
36              //его обработка методами класса
37              Library<Number> obj = new Library<Number>(vector);
38              System.out.println(reportPart1);
39              //Обработка одномерного вектора
40              System.out.println("Инициализированный вектор");
41              System.out.println(obj.vectorToString());
42              obj.min();
43              System.out.println("Минимум в векторе: "
44                                 + obj.getMin());
45              //Интерпретация одномерного вектора как матрицы 4x4
46              System.out.println(reportPart2);
47              WorkWithFile.writeMatrixTXT(vector, "matrix.txt");
48              System.out.println("Количество строк в файле: " +
49                                 WorkWithFile
50                                 .numberStringInTXTFile("matrix.txt")
51              //построчное чтение файла и преобразование в
52              //одномерную коллекцию
53              obj = new Library<Number>(WorkWithFile
54                                        .readMatrixTXT("matrix.txt"));
55              System.out.println("Матрица:");
56              System.out.println(obj.matrixToString());
57              obj.maxDiagonalMatrix();
58              System.out.println("Максимум на диагонали матрицы: " +
59                                 obj.getMax());
60          }
61          catch(IOException ex) {
62              System.out.println(ex.getMessage());
63          }
64      }
65  }
```

Результат работы программы:

```
input
Введите значение НИЖНЕЙ границы генерации чисел в коллекции:
-15
Введите значение ВЕРХНЕЙ границы генерации чисел в коллекции:
15

--ОДНОМЕРНЫЙ МАССИВ

Инициализированный вектор
-6 -8 -5 14 8 1 2 10 -3 -9 -8 -5 13 -8 -9 -3
Минимум в векторе: -9

--ИНТЕРПРЕТАЦИЯ ВЕКТОРА КАК МАТРИЦЫ

Количество строк в файле: 4
Матрица:
  -6   -8   -5   14
   8    1    2   10
  -3   -9   -8   -5
  13   -8   -9   -3
Максимум на диагонали матрицы: 1
```

```
//содержимое файла vector.txt
```

Main.java | InitCollection.java | Library.java | StringToNumber.java | WorkWithFile.java | vector.txt | matrix.txt

```
1  -6 -8 -5 14 8 1 2 10 -3 -9 -8 -5 13 -8 -9 -3
```

```
//содержимое файла matrix.txt
```

Main.java | InitCollection.java | Library.java | StringToNumber.java | WorkWithFile.java | vector.txt | matrix.txt

```
1  -6 -8 -5 14
2  8 1 2 10
3  -3 -9 -8 -5
4  13 -8 -9 -3
```

7.2. Индивидуальные задания

7.2.1. Часть 1

Обработка одномерных коллекций размерностью $\boldsymbol{n}$ (где $\sqrt{\boldsymbol{n}}$ – целое число). В качестве свойств класса `Library` использовать две коллекции с именами `x` и `y` каждая по `n` чисел со случайными значениями.

Замечание.
Обе коллекции должны содержать как положительные, так и отрицательные значения.

Понимая, что элементы коллекций `x[i]`, `y[i]` обозначают координаты точек на плоскости и любые три точки определяют треугольник, вычислить:

1. определить ***наименьшую*** площадь среди ***всех*** подобных треугольников;
2. определить ***наибольший*** радиус вписанной окружности для треугольников, имеющих вершины в точках с ***нечетными*** индексами.
3. определить ***наименьший*** периметр среди ***всех*** подобных треугольников;
4. определить ***наименьший*** радиус описанной окружности для треугольников, имеющих вершины в точках с ***четными*** индексами;
5. определить ***максимальную*** площадь среди ***всех*** подобных треугольников;
6. определить ***наименьший*** радиус вписанной окружности для треугольников, имеющих вершины в точках с ***четными*** индексами;

7. определить ***наибольший*** периметр среди ***всех*** подобных треугольников;
8. определить ***наибольший*** радиус описанной окружности для треугольников, имеющих вершины в точках с ***нечетными*** индексами;
9. определить ***наименьшую*** площадь треугольников, имеющих вершины в точках с ***четными*** индексами;
10. определить ***наибольший*** радиус вписанной окружности среди ***всех*** подобных треугольников;
11. определить ***наименьший*** периметр треугольников, имеющих вершины в точках с ***четными*** индексами;
12. определить ***максимальную*** площадь треугольников, имеющих вершины в точках с ***нечетными*** индексами;
13. определить ***наибольший*** радиус описанной окружности среди ***всех*** подобных треугольников;
14. определить ***наибольший*** периметр треугольников, имеющих вершины в точках с ***нечетными*** индексами;
15. определить ***наименьший*** радиус вписанной окружности среди ***всех*** подобных треугольников;
16. определить ***наименьший*** радиус описанной окружности среди ***всех*** подобных треугольников;

7.2.2. Часть 2

Интерпретация одномерной коллекции из части 1 размерностью $\boldsymbol{n}$ (либо `x[i]` либо `y[j]` ***по выбору разработчика***) как квадратного двумерного массива с размерностью $\sqrt{\boldsymbol{n}} \times \sqrt{\boldsymbol{n}}$.

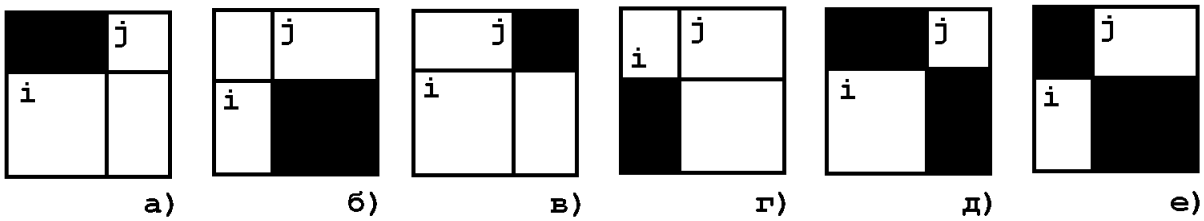
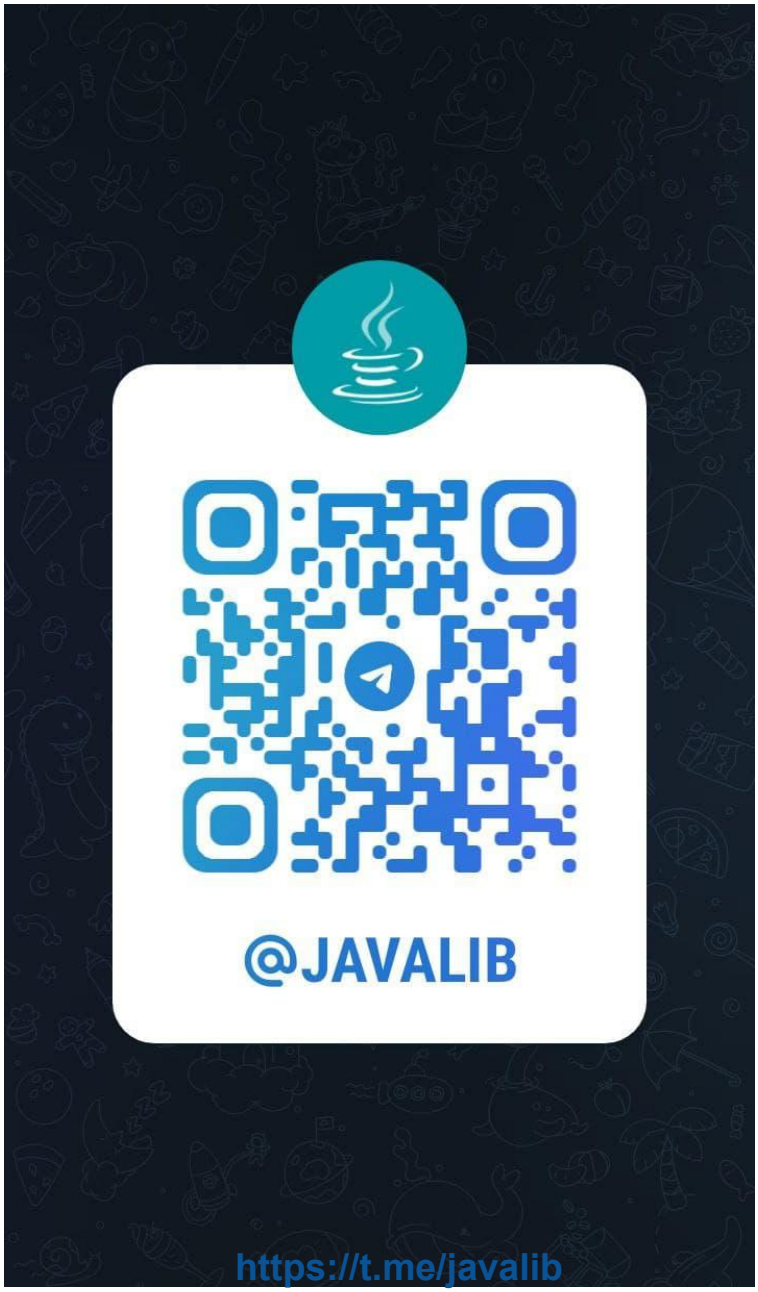


Рисунок 29 – Вид обрабатываемых областей в двумерном массиве

1. определить наи***большее*** среднее арифметическое значение элементов всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, а);
2. определить наи***большее*** среднее арифметическое значение элементов с ***четными*** индексами ***столбцов*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, е);

3. определить наи***меньшее*** среднее арифметическое значение элементов с ***нечетными*** индексами ***строк*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, б);
4. определить наи***меньшее*** среднее арифметическое значение элементов всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, д);
5. определить наи***большее*** среднее арифметическое значение элементов с ***четными*** индексами ***столбцов*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, в);
6. определить наи***меньшее*** среднее арифметическое значение элементов всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, б);
7. определить наи***меньшее*** среднее арифметическое значение элементов с ***нечетными*** индексами ***столбцов*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, г);
8. определить наи***большее*** среднее арифметическое значение элементов всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, е);
9. определить наи***большее*** среднее арифметическое значение элементов с ***четными*** индексами ***строк*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, а);
10. определить наи***большее*** среднее арифметическое значение элементов в подобластях двумерного массива, указанных на рисунке (Рисунок 29, в);
11. определить ***наибольшее*** среднее арифметическое значение элементов с ***нечетными*** индексами ***строк*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, а);
12. определить наи***меньшее*** среднее арифметическое значение элементов с ***четными*** индексами ***столбцов*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, г);
13. определить наи***большее*** среднее арифметическое значение элементов с ***нечетными*** индексами ***столбцов*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1]$, $[0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, в);

14. определить наи***меньшее*** среднее арифметическое значение элементов с ***четными*** индексами ***строк*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1], [0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, д);
15. определить наи***меньшее*** среднее арифметическое значение элементов всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1], [0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, г);
16. определить наи***меньшее*** среднее арифметическое значение элементов с ***четными*** индексами ***строк*** всех возможных подобластей двумерного массива ($i = [0, \sqrt{n} - 1], [0, \sqrt{n} - 1]$), указанных на рисунке (Рисунок 29, б);



Литература

1. Блинов, И.Н. Java from EPAM / И.Н. Блинов, В.С. Романчик. - Минск: Четыре четверти, 2020. - 560 с.
2. Кравчук, А.И. Сборник лабораторных работ и примеров решения задач по алгоритмизации и программированию на языке Си / А.И. Кравчук, А.С. Кравчук - Минск: Технопринт, 2002. - 116 с.
3. Информатика и программирование: лабораторный практикум по программированию на языке С++ : учеб.-метод. пособие / В.Ю. Наумов, О.А. Авдеюк; ВолгГТУ. – 2-е изд., испр. – Волгоград, 2018. – 136 с.
4. Практическое руководство по языку С++ / Н.А. Аленский– Минск: АПО, 2007. – 276 с.
5. Методы программирования: лекции, примеры, тесты. В 2 ч. Ч. 1. / Н.А. Аленский – Минск: БГУ, 2012. – 87 с.
6. Игнатьева, О.В. Информатика и программирование. В 2 ч. Ч. 1 / О.В. Игнатьева; ФГБОУ ВО РГУПС. – Ростовн/Д, 2017. – 205 с.
7. Язык Java. Основы синтаксиса / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/290065 (Дата доступа: 11.03.2023)
8. Язык Java. Операторы управления программой / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/290066 (Дата доступа: 11.03.2023)
9. Язык Java. Массивы, строки, классы обертки базовых типов / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/290067 (Дата доступа: 11.03.2023)
10. Язык Java. Методы, стек и куча / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/290068 (Дата доступа: 11.03.2023)
11. Язык Java. Дженерики : учеб. материалы для студентов спец. 1-31 03 08 «Математика и информационные технологии (по направлениям)» / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/294803 (Дата доступа: 11.03.2023)
12. Язык Java. Общие сведения о коллекциях и реализуемых ими интерфейсах. Коллекции, реализующие интерфейс List / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/291422 (Дата доступа: 11.03.2023)
13. Язык Java. Коллекции, реализующие интерфейсы Queue, Deque, Map, Set / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/291423 (Дата доступа: 11.03.2023)

14. Язык Java. Stream API : учеб. материалы для студентов спец. 1-31 03 08 «Математика и информационные технологии (по направлениям)» / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/294802 (Дата доступа: 11.03.2023)
15. Язык Java. Потоки ввода-вывода. Работа с файлами : учеб. материалы для студентов спец. 1-31 03 08 «Математика и информационные технологии (по направлениям)» / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/294799 (Дата доступа: 11.03.2023)
16. Язык Java. Лямбда-выражения : учеб. материалы для студентов спец. 1-31 03 08 «Математика и информационные технологии (по направлениям)» / А.С. Кравчук, А.И. Кравчук, Е.В. Кремень [Электронный документ] URL: https://elib.bsu.by/handle/123456789/294798 (Дата доступа: 11.03.2023)