МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет имени академика С.П. Королева»
(Самарский университет)

ОТЧЕТ ПО
ЛАБОРАТОРНОЙ РАБОТЕ № 3

«Обработка исключений и разработка классов табулированной функции»

по курсу
Объектно-ориентированное программирование

Выполнила: Грачева Виктория,
студентка группы 6203-010302D

Задание 1

Для выполнения лабораторной работы я изучила следующие классы исключений Java:

- java.lang.Exception
- java.lang.IndexOutOfBoundsException
- java.lang.ArrayIndexOutOfBoundsException
- java.lang.IllegalArgumentException
- java.lang.IllegalStateException

Задание 2

В пакете functions были созданы два класса исключений:
FunctionPointIndexOutOfBoundsException - класс, наследующий от IndexOutOfBoundsException и выбрасывающий исключение выхода за границы точек при обращении к ним по номеру;

InappopriateFunctionPointException - класс, наследующий от Exception и выбрасывающий исключение при попытке добавления или изменения точки функции несоответствующим образом;

```java
package functions;

public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException {  35 usages
    public FunctionPointIndexOutOfBoundsException() { super(); }
    public FunctionPointIndexOutOfBoundsException(String msg) { super(msg); }
    public FunctionPointIndexOutOfBoundsException(int number) { super("Набор точек выходит за границы: "+ number); }
}
```

```java
package functions;

public class InappropriateFunctionPointException extends Exception {  21 usages
    public InappropriateFunctionPointException() { super(); }
    public InappropriateFunctionPointException(String msg) { super(msg); }
}
```

При этом, в классах используется ключевое слово super() - вызов конструктора родительского класса (суперкласса). Это позволяет инициализировать унаследованные поля и поведение.

Задание 3

В этом задании мы модифицируем уже существующий класс TabulatedFunction:

- Теперь конструкторы выбрасывают исключение IllegalArgumentException, если введены неверные параметры;
- Все методы, используемые для работы с точками, содержат исключение FunctionPointIndexOutOfBoundsException, которое выводится при некорректных индексах;
- setPoint(), setPointX(), addPoint() также применяют исключение InappopriateFunctionPointException, если нарушен порядок точек;
- В метод удаления точки deletePoint() внесено IllegalStateException, если мы пытаемся удалить точку при количестве точек, меньшем трёх.

Ниже я представила некоторые из этих модификаций:

```
public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount)  5 usages
        throws IllegalArgumentException {
        if (leftX >= rightX) {
            throw new IllegalArgumentException("Левая граница области определения больше или равна правой");
        }
        if (pointsCount < 2) {
            throw new IllegalArgumentException("Количество точек меньше 2");
        }

    this.pointsCount = pointsCount;
    this.points = new FunctionPoint[pointsCount + 10];

    double step = (rightX - leftX) / (pointsCount - 1);

    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + i * step;
        points[i] = new FunctionPoint(x,  y: 0);
    }
}
```

(скриншоты сделаны уже после выполнения всех заданий, поэтому TabulatedFunction уже переименован)

```java
public void setPoint(int index, FunctionPoint point)   no usages
        throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException {
    if (index<0 || index >= pointsCount){
        throw new FunctionPointIndexOutOfBoundsException(index);
    }

    // проверяем границы для новой точки
    if (index > 0 && (point.getter_x() < points[index - 1].getter_x()
            ||Comparison(point.getter_x(),points[index - 1].getter_x()))) {
        throw new InappropriateFunctionPointException("X лежит вне определенного интервала");
    }
    if (index < pointsCount - 1 && (point.getter_x() > points[index + 1].getter_x())
            ||Comparison(point.getter_x(),points[index + 1].getter_x())) {
        throw new InappropriateFunctionPointException("X лежит вне определенного интервала");
    }

    // копия точки
    points[index] = new FunctionPoint(point);
}
```

```java
//методы изменения количества точек
public void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException {  1 usage
    if (index<0 || index>=pointsCount){
        throw new FunctionPointIndexOutOfBoundsException(index);
    }
    if(pointsCount<3){
        throw new IllegalStateException("Количество точек меньше 3");
    }
    System.arraycopy(points, srcPos: index + 1, points, index, length: pointsCount - index - 1);
    pointsCount--;
}
```

Задания 4-5

В текущих заданиях я реализовала класс LinkedListTabulatedFunction, неeобходимый для описания двусвязного циклического списка и работы с ним, а также описания работы с табулированной функцией и ее точками.

Для этого было сделано следующее:

- Создан внутренний класс FunctionNode для элементов списка;
- Реализованы методы getNodeByIndex(), addNodeToTail(), addNodeByIndex(), deleteNodeByIndex() для дальнейшей работы со списком;
- Полная реализация интерфейса TabulatedFunction;
- Обработка всех необходимых исключений.

Далее представлены некоторые упомянутые реализованные фрагменты класса LinkedListTabulatedFunction:

```
public class LinkedListTabulatedFunction implements TabulatedFunction

    // Внутренний класс для узла списка
    private static class FunctionNode {  24 usages
        FunctionPoint point;  19 usages
        FunctionNode last;  13 usages
        FunctionNode next;  21 usages

        FunctionNode(FunctionPoint point) { this.point = point; }
    }
```

```
 //Методы интерфейса TabulatedFunction

 public int getPointsCount() { return size; }

 public FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException {  1 usage
     return new FunctionPoint(getNodeByIndex(index).point);
 }

 public void setPoint(int index, FunctionPoint point)  no usages
         throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException {
     FunctionNode node = getNodeByIndex(index);

     // Проверка порядка X
     if (index > 0 && point.getter_x() <= node.last.point.getter_x()) {
         throw new InappropriateFunctionPointException("Новый x должен быть между соседними точками");
     }
     if (index < size - 1 && point.getter_x() >= node.next.point.getter_x()) {
         throw new InappropriateFunctionPointException("Новый x должен быть между соседними точками");
     }

     node.point = new FunctionPoint(point);
 }
```

```
// Добавление узла в конец списка
private FunctionNode addNodeToTail(FunctionPoint point) {  3 usages
    FunctionNode newNode = new FunctionNode(point);
    FunctionNode tail = head.last;

    newNode.last = tail;
    newNode.next = head;
    tail.next = newNode;
    head.last = newNode;

    size++;
    lastReadWriteNode = newNode;
    lastReadWriteIndex = size - 1;

    return newNode;
}
```

Задание 6

На этом этапе класс TabulatedFunction был переименован в ArrayTabulatedFunction, создан интерфейс TabulatedFunction, в котором содержатся объявления общих методов классов ArrayTabulatedFunction и LinkedListTabulatedFunction(getPoint(), addPoint(), getPointX, getPointsCount и т. п).

```
package functions;

public interface TabulatedFunction {  12 usages  2 implementations

    int getPointsCount();  4 usages  2 implementations

    FunctionPoint getPoint(int index)  1 usage  2 implementations
            throws FunctionPointIndexOutOfBoundsException;

    void setPoint(int index, FunctionPoint point)  no usages  2 implementations
            throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;

    double getPointX(int index)  3 usages  2 implementations
            throws FunctionPointIndexOutOfBoundsException;

    void setPointX(int index, double x)  1 usage  2 implementations
            throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;

    double getPointY(int index)  1 usage  2 implementations
            throws FunctionPointIndexOutOfBoundsException;


    void setPointY(int index, double y)  2 usages  2 implementations
            throws FunctionPointIndexOutOfBoundsException;

    void deletePoint(int index)  1 usage  2 implementations
            throws FunctionPointIndexOutOfBoundsException, IllegalStateException;

    void addPoint(FunctionPoint point)  1 usage  2 implementations
            throws InappropriateFunctionPointException;
```


ArrayTabulatedFunction и LinkedListTabulatedFunction реализуют интерфейс, это я указала в объявлении этих классов, добавив ключевое слово implements.

Задание 7

Класс Main был модифицирован и теперь проводит тестирование всех функций и обработку исключений. В тесте ArrayTabulatedFunction создает функцию синус, а Тест LinkedListTabulatedFunction косинус.

** Тест ArrayTabulatedFunction **

Array функция создана успешно

Точки функции: (0,00, 0,00) (2,50, 0,60) (5,00, -0,96) (7,50, 0,94) (10,00, -0,54)

** Тест LinkedListTabulatedFunction **

LinkedList функция создана успешно

Точки функции: (-5,00, 0,28) (-3,00, -0,99) (-1,00, 0,54) (1,00, 0,54) (3,00, -0,99) (5,00, 0,28)

** Тестирование исключений **

** Тест исключений конструкторов:

Левая граница области определения больше или равна правой

Количество точек меньше 2

** Тестирование исключений доступа по индексу:

Набор точек выходит за границы: 10

** Тестирование исключений при изменении точек:

Новый x должен быть между соседними точками

** Тестирование исключений при добавлении точек:

Точка с такой координатой x уже есть

** Тестирование исключений при удалении точек:

Нельзя удалить точку, потому что точек меньше 3

** Тестирование полиморфизма:

Функция 1 типа: ArrayTabulatedFunction

Количество точек: 3

Границы: [0.0, 5.0]

Функция 2 типа: LinkedListTabulatedFunction

Количество точек: 3

Границы: [0.0, 5.0]

Process finished with exit code 0