

Министерство науки и высшего образования Российской Федерации

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

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №3

Дисциплина: «ООП»

Выполнил: Илларионов Е.А.

Группа: 6201-120303D

Самара, 2025

Задание 1

Изучил классы исключений Java:

java.lang.Exception

java.lang.IndexOutOfBoundsException

java.lang.ArrayIndexOutOfBoundsException

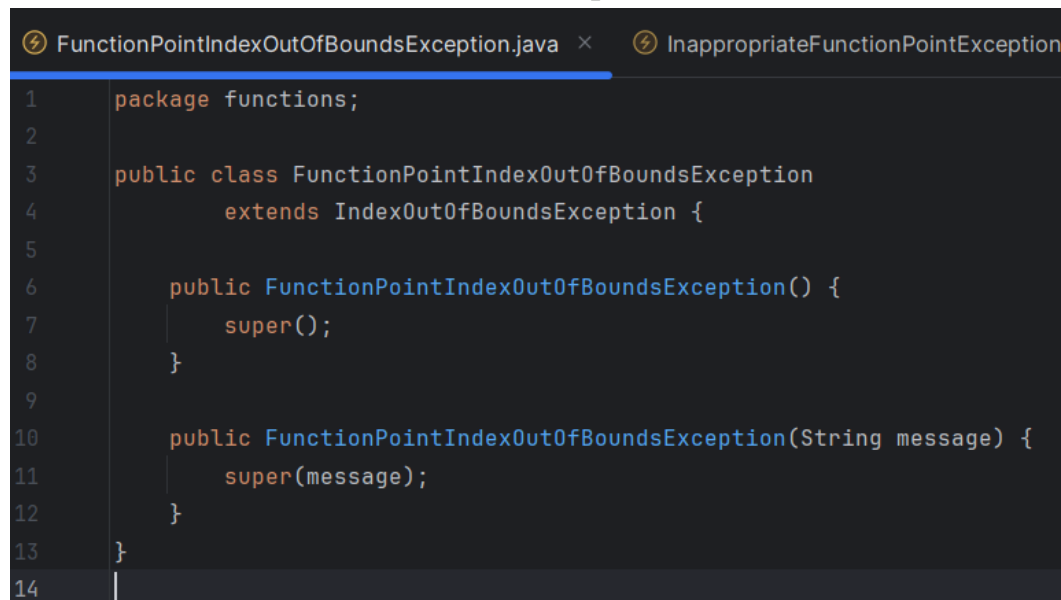
java.lang.IllegalArgumentException

java.lang.IllegalStateException

Задание 2

В пакете functions создал 2 класса исключений

FunctionPointIndexOutOfBoundsException



```
1 package functions;
2
3 public class FunctionPointIndexOutOfBoundsException
4     extends IndexOutOfBoundsException {
5
6     public FunctionPointIndexOutOfBoundsException() {
7         super();
8     }
9
10    public FunctionPointIndexOutOfBoundsException(String message) {
11        super(message);
12    }
13 }
14
```

InappropriateFunctionPointException



```
1 package functions;
2
3 public class InappropriateFunctionPointException extends Exception {
4
5     public InappropriateFunctionPointException() {
6         super();
7     }
8
9     public InappropriateFunctionPointException(String message) {
10        super(message);
11    }
12 }
13
```

Задание 3

В разработанный ранее класс `TabulatedFunction` внес изменения, обеспечивающие выбрасывание исключений методом класса

```
11 public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
12     if (leftX >= rightX) {
13         throw new IllegalArgumentException("Левая граница должна быть строго меньше правой.");
14     }
15     if (pointsCount < 2) {
16         throw new IllegalArgumentException("Количество точек должно быть >= 2.");
17     }
18
19     this.size = pointsCount;
20     this.points = new FunctionPoint[pointsCount];
21     double step = (rightX - leftX) / (pointsCount - 1);
22
23     for (int i = 0; i < pointsCount; i++) {
24         double x = leftX + i * step;
25         this.points[i] = new FunctionPoint(x, 0);
26     }
27 }
```

```
29 public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {
30     if (leftX >= rightX) {
31         throw new IllegalArgumentException("Левая граница должна быть строго меньше правой.");
32     }
33     if (values.length < 2) {
34         throw new IllegalArgumentException("Массив значений должен содержать >= 2 элементов.");
35     }
36
37     this.size = values.length;
38     this.points = new FunctionPoint[size];
39     double step = (rightX - leftX) / (size - 1);
40
41     for (int i = 0; i < size; i++) {
42         double x = leftX + i * step;
43         this.points[i] = new FunctionPoint(x, values[i]);
44     }
45 }
```

Методы `getPoint()`, `getPointX()`, `getPointY()`, `setPointY()` выбрасывают исключение `FunctionPointIndexOutOfBoundsException`, если переданный в метод номер выходит за границы набора точек.

Методы `setPoint()`, `setPointX()`, выбрасывают исключение `FunctionPointIndexOutOfBoundsException`, если переданный в метод номер выходит за границы набора точек, и исключение `InappropriateFunctionPointException` в том случае, если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции.

Задание 4

В пакете `functions` создал класс `LinkedListTabulatedFunction`.

Класс `FunctionNode` реализован как приватный вложенный класс, обеспечивая инкапсуляцию структуры списка.

```
3      public class LinkedListTabulatedFunction implements TabulatedFunction {
4
5          // Вложенный приватный класс FunctionNode
6          private static class FunctionNode {
7              public FunctionPoint point;
8              public FunctionNode prev;
9              public FunctionNode next;
10
11              public FunctionNode(FunctionPoint point) {
12                  this.point = point;
13                  this.prev = this.next = this;
14              }
15          }
16      }
```

Реализован метод `getNodeByIndex(int index)`, который использует кэширование последнего узла и выбирает направление поиска (от начала или от конца списка) для ускорения операций.

```
74      private FunctionNode getNodeByIndex(int index) {
75          FunctionNode current;
76
77          if (index == lastAccessedIndex) {
78              return lastAccessedNode;
79          }
80
81          if (index < size / 2) {
82              current = head.next;
83              for (int i = 0; i < index; i++) {
84                  current = current.next;
85              }
86          } else {
87              current = head.prev;
88              for (int i = size - 1; i > index; i--) {
89                  current = current.prev;
90              }
91          }
92      }
```

Задание 5

В классе `LinkedListTabulatedFunction` реализованы публичные конструкторы и методы, соответствующие сигнатурам `ArrayTabulatedFunction`

```

56     public LinkedListTabulatedFunction(double leftX, double rightX, double[] values) {
57         if (leftX >= rightX) {
58             throw new IllegalArgumentException("Левая граница должна быть строго меньше правой.");
59         }
60         if (values.length < 2) {
61             throw new IllegalArgumentException("Массив значений должен содержать >= 2 элементов.");
62         }
63
64         initializeList();
65
66         double step = (rightX - leftX) / (values.length - 1);
67
68         for (int i = 0; i < values.length; i++) {
69             double x = leftX + i * step;
70             addNodeToTail(new FunctionPoint(x, values[i]));
71         }
72     }

```

```

166     public double getRightDomainBorder() {
167         if (size == 0) return Double.NaN;
168         return head.prev.point.getX();
169     }
170
171     @Override
172     public double getFunctionValue(double x) {
173         if (x < getLeftDomainBorder() || x > getRightDomainBorder()) return Double.NaN;
174
175         FunctionNode current = head.next;
176         if (lastAccessedIndex != -1) {
177             current = lastAccessedNode;
178         }
179
180         while (current.next != head) {
181             FunctionNode nextNode = current.next;
182
183             if (x >= current.point.getX() && x <= nextNode.point.getX() + EPSILON) {
184                 lastAccessedNode = current;
185                 lastAccessedIndex = -1;
186                 // Линейная интерполяция
187                 double x0 = current.point.getX();
188                 double y0 = current.point.getY();
189                 double x1 = nextNode.point.getX();
190                 double y1 = nextNode.point.getY();
191
192                 if (Math.abs(x1 - x0) < EPSILON) return y0;
193
194                 return y0 + (y1 - y0) * (x - x0) / (x1 - x0);
195             }

```

```

196         current = current.next;
197     }
198     return Double.NaN;
199 }
200
201 @Override
202 public int getPointsCount() {
203     return size;
204 }
205
206 @Override
207 public FunctionPoint getPoint(int index) {
208     checkIndex(index);
209     return new FunctionPoint(getNodeByIndex(index).point);
210 }
211
212 @Override
213 public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException {
214     checkIndex(index);
215
216     FunctionNode targetNode = getNodeByIndex(index);
217     double newX = point.getX();
218
219     if (targetNode.prev != head && newX < targetNode.prev.point.getX() + EPSILON) {
220         throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго больше предыду");
221     }
222     if (targetNode.next != head && newX > targetNode.next.point.getX() - EPSILON) {
223         throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго меньше следующ");
224     }
225
226     targetNode.point.setX(newX);
227     targetNode.point.setY(point.getY());
228 }
229
230 @Override
231 public double getPointX(int index) {
232     checkIndex(index);
233     return getNodeByIndex(index).point.getX();
234 }
235
236 @Override
237 public void setPointX(int index, double x) throws InappropriateFunctionPointException {
238     checkIndex(index);
239
240     FunctionNode targetNode = getNodeByIndex(index);

```

```

241
242         if (targetNode.prev != head && x < targetNode.prev.point.getX() + EPSILON) {
243             throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго бо
244         }
245         if (targetNode.next != head && x > targetNode.next.point.getX() - EPSILON) {
246             throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго ме
247         }
248         targetNode.point.setX(x);
249     }
250
251     @Override
252     public double getPointY(int index) {
253         checkIndex(index);
254         return getNodeByIndex(index).point.getY();
255     }
256
257     @Override
258     public void setPointY(int index, double y) {
259         checkIndex(index);
260         getNodeByIndex(index).point.setY(y);
261     }
262
263     @Override
264     public void deletePoint(int index) {
265         if (size <= 2) {
266             throw new IllegalStateException("Невозможно удалить точку: количество точек должно быть не мен
267         }
268
269         checkIndex(index);
270         deleteNodeByIndex(index);
271     }
272
273     @Override
274     public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {
275
276         FunctionNode current = head.next;
277         int pos = 0;
278
279         while (current != head) {
280             if (Math.abs(current.point.getX() - point.getX()) < EPSILON) {
281                 throw new InappropriateFunctionPointException("Точка с такой абсциссой уже существует: " +
282             }
283             if (current.point.getX() > point.getX()) {
284                 break;
285             }
286             current = current.next;
287             pos++;
288         }
289
290         addNodeByIndex(pos, point);
291     }
292

```

Задание 6

Класс TabulatedFunction переименован в ArrayTabulatedFunction.

Создан интерфейс TabulatedFunction.java, содержащий объявления всех общих методов.

Оба класса функций (ArrayTabulatedFunction и LinkedListTabulatedFunction) реализовали созданный интерфейс.

Задание 7

Класс main

```
1  import functions.FunctionPoint;
2  import functions.TabulatedFunction;
3  import functions.ArrayTabulatedFunction;
4  import functions.LinkedListTabulatedFunction;
5  import functions.FunctionPointIndexOutOfBoundsException;
6  import functions.InappropriateFunctionPointException;
7
8  public class Main {
9
10     // Проверка исключений
11     public static void testExceptions(TabulatedFunction func, String name) {
12         System.out.println("\n--- Тестирование исключений для " + name + " ---");
13
14         // 1. IllegalArgumentException – конструктор
15         try {
16             new ArrayTabulatedFunction( leftX: 10, rightX: 0, pointsCount: 5);
17         } catch (IllegalArgumentException e) {
18             System.out.println("Конструктор (OK): " + e.getMessage());
19         }
20
21         // 2. IllegalStateException – deletePoint при size <= 2
22         try {
23             double[] values = {1, 2};
24             TabulatedFunction shortFunc = new ArrayTabulatedFunction( leftX: 0, rightX: 1, values);
25             shortFunc.deletePoint( index: 0);
26         } catch (IllegalStateException e) {
27             System.out.println("deletePoint (OK): " + e.getMessage());
28         }
29
30         // 3. FunctionPointIndexOutOfBoundsException
31         try {
```



```

32         func.getPoint( index: 100);
33     } catch (FunctionPointIndexOutOfBoundsException e) {
34         System.out.println("getPoint (OK): " + e.getMessage());
35     }
36
37     // 4. InappropriateFunctionPointException - addPoint
38     try {
39         func.addPoint(new FunctionPoint(func.getPointX( index: 1), y: 50));
40     } catch (InappropriateFunctionPointException e) {
41         System.out.println("addPoint (OK): " + e.getMessage());
42     }
43
44     // 5. InappropriateFunctionPointException - setPointX
45     try {
46         func.setPointX( index: 1, x: func.getPointX( index: 0) - 1);
47     } catch (InappropriateFunctionPointException e) {
48         System.out.println("setPointX (OK): " + e.getMessage());
49     }
50
51     System.out.println("Тестирование исключений завершено.");
52 }
53
54 // Проверка добавления, удаления и замены нулевой точки
55 public static void testZeroIndexOperations(TabulatedFunction func, String name) {
56     System.out.println("\n--- Проверка операций с нулевой точкой: " + name + " ---");
57
58     // 1. Замена нулевой точки
59     try {
60         FunctionPoint oldPoint = func.getPoint( index: 0);
61         FunctionPoint newPoint =
62             new FunctionPoint(oldPoint.getX(), y: oldPoint.getY() + 100);
63
64         func.setPoint( index: 0, newPoint);
65         System.out.println("setPoint(0) OK: y = " + func.getPointY( index: 0));
66     } catch (Exception e) {
67         System.out.println("setPoint(0) ERROR: " + e.getMessage());
68     }
69
70     // 2. Удаление нулевой точки
71     try {
72         int before = func.getPointsCount();
73         func.deletePoint( index: 0);
74         int after = func.getPointsCount();
75
76         System.out.println("deletePoint(0) OK: " + before + " -> " + after);
77         System.out.println("Новая первая точка: x = " + func.getPointX( index: 0));
78     } catch (Exception e) {
79         System.out.println("deletePoint(0) ERROR: " + e.getMessage());
80     }
81
82     // 3. Добавление новой первой точки
83     try {
84         double newX = func.getLeftDomainBorder() - 1;
85         FunctionPoint newPoint = new FunctionPoint(newX, y: 999);
86

```

```

87         func.addPoint(newPoint);
88         System.out.println("addPoint(new first) OK: x = " + func.getPointX(index: 0));
89     } catch (Exception e) {
90         System.out.println("addPoint(new first) ERROR: " + e.getMessage());
91     }
92 }
93
94 public static void main(String[] args) {
95
96     double[] values = {10, 20, 30, 40, 50};
97
98     // Проверка обычной работы
99     TabulatedFunction arrayFunc =
100         new ArrayTabulatedFunction(leftX: 0, rightX: 4, values);
101     TabulatedFunction linkedFunc =
102         new LinkedListTabulatedFunction(leftX: 0, rightX: 4, values);
103
104     System.out.println("--- Проверка работы ArrayTabulatedFunction ---");
105     System.out.println("f(1.5) = " + arrayFunc.getFunctionValue(x: 1.5));
106
107     System.out.println("\n--- Проверка работы LinkedListTabulatedFunction ---");
108     System.out.println("Количество точек: " + linkedFunc.getPointsCount());
109     System.out.println("Правая граница: " + linkedFunc.getRightDomainBorder());
110     System.out.println("f(2.5) = " + linkedFunc.getFunctionValue(x: 2.5));
111
112     //Проверка исключений
113     testExceptions(arrayFunc, name: "ArrayTabulatedFunction");
114
115     testExceptions(linkedFunc, name: "LinkedListTabulatedFunction");
116
117     //Проверка нулевой точки
118     testZeroIndexOperations(
119         new ArrayTabulatedFunction(leftX: 0, rightX: 4, new double[]{10, 20, 30, 40}),
120         name: "ArrayTabulatedFunction"
121     );
122
123     testZeroIndexOperations(
124         new LinkedListTabulatedFunction(leftX: 0, rightX: 4, new double[]{10, 20, 30, 40}),
125         name: "LinkedListTabulatedFunction"
126     );
127 }
128

```

Результат

```
--- Проверка работы ArrayTabulatedFunction ---
f(1.5) = 25.0

--- Проверка работы LinkedListTabulatedFunction ---
Количество точек: 5
Правая граница: 4.0
f(2.5) = 35.0

--- Тестирование исключений для ArrayTabulatedFunction ---
Конструктор (OK): Левая граница должна быть строго меньше правой.
deletePoint (OK): Невозможно удалить точку: количество точек должно быть не менее трех.
getPoint (OK): Индекс 100 выходит за границы [0, 4]
addPoint (OK): Точка с такой абсциссой уже существует: 1.0
setPointX (OK): Новая абсцисса должна быть строго больше предыдущей точки.
Тестирование исключений завершено.

--- Тестирование исключений для LinkedListTabulatedFunction ---
Конструктор (OK): Левая граница должна быть строго меньше правой.
deletePoint (OK): Невозможно удалить точку: количество точек должно быть не менее трех.
getPoint (OK): Индекс 100 выходит за границы [0, 4]
addPoint (OK): Точка с такой абсциссой уже существует: 1.0
setPointX (OK): Новая абсцисса должна быть строго больше предыдущей точки.
Тестирование исключений завершено.
```

```
--- Проверка операций с нулевой точкой: ArrayTabulatedFunction ---
setPoint(0) OK: y = 110.0
deletePoint(0) OK: 4 -> 3
Новая первая точка: x = 1.3333333333333333
addPoint(new first) OK: x = 0.33333333333333326

--- Проверка операций с нулевой точкой: LinkedListTabulatedFunction ---
setPoint(0) OK: y = 110.0
deletePoint(0) OK: 4 -> 3
Новая первая точка: x = 1.3333333333333333
addPoint(new first) OK: x = 0.33333333333333326
```