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

«Работа с исключениями и интерфейсами в наборе классов табулированной функции»

по курсу
Объектно-ориентированное программирование

Выполнила: Гонтарь Анастасия Вячеславовна,
студент группы 6203-010302D

Содержание

Задание №1

Я ознакомилась с документацией с классами исключений, входящих в API Java

Задание №2

В пакете functions я создала два класса исключений:

• FunctionPointIndexOutOfBoundsException – исключение выхода за границы набора точек при обращении к ним по номеру, наследует от класса IndexOutOfBoundsException;

```
package functions;

public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException{  30 usages
    public FunctionPointIndexOutOfBoundsException(){  6 usages
        super();
    }

    public FunctionPointIndexOutOfBoundsException(String message) { super(message); }
}
```

• InappropriateFunctionPointException – исключение, выбрасываемое при попытке добавления или изменения точки функции несоответствующим образом, наследует от класса Exception.

```
package functions;

public class InappropriateFunctionPointException extends Exception{  17 usages
    public InappropriateFunctionPointException (){  8 usages
        super();
    }

    public InappropriateFunctionPointException (String message) { super(message); }
}
```

Задание №3

В разработанный ранее класс TabulatedFunction я внесла изменения, обеспечивающие выбрасывание исключений методами класса.

• Оба конструктора класса должны выбрасывать исключение IllegalArgumentException, если левая граница области определения больше или равна правой, а также если предлагаемое количество точек меньше двух

```java
// создание табулированной функции
public TabulatedFunction(double leftX, double rightX, int pointsCount){  13 usages
    if (leftX>=rightX)
        throw new IllegalArgumentException("Левая граница больше правой");
    if (pointsCount <2)
        throw new IllegalArgumentException("Количество точек меньше двух");
    this.pointsCount = pointsCount;
    points = new FunctionPoint[pointsCount];
    double step = Math.abs(rightX-leftX)/(pointsCount-1);
    for (int i=0; i < pointsCount;i++) {
        points[i] = new FunctionPoint( x: leftX + i*step,  y: 0);
    }
}

// создание табулированной функции с заданными значениями по оси ординат
public TabulatedFunction(double leftX, double rightX, double[] value){  12 usages
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница больше правой");
    }
    if (value.length < 2) {
        throw new IllegalArgumentException("Количество точек меньше двух");
    }
    int pointsCount = value.length;
    this.pointsCount = pointsCount;
    points = new FunctionPoint[pointsCount];
    double step = Math.abs(rightX-leftX)/(pointsCount-1);
    for (int i=0; i < pointsCount;i++) {
        points[i] = new FunctionPoint( x: leftX + i*step, value[i]);
    }
}
```

• Методы getPoint(), setPoint(), getPointX(), setPointX(), getPointY(), setPointY() и deletePoint() должны выбрасывать исключение FunctionPointIndexOutOfBoundsException, если переданный в метод номер выходит за границы набора точек.

• Методы setPoint() и setPointX() должны выбрасывать исключение InappropriateFunctionPointException в том случае, если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции. Метод addPoint() также должен выбрасывать исключение InappropriateFunctionPointException, если в наборе точек функции есть точка, абсцисса которой совпадает с абсциссой добавляемой точки.

- Метод deletePoint() должен выбрасывать исключение IllegalStateException, если на момент удаления точки количество точек в наборе менее трех.

```java
// возвращение точки
public FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException {  no usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    return new FunctionPoint(points[index]);
}

// изменение указанной точки на заданную
public void setPoint (int index, FunctionPoint point) throws InappropriateFunctionPointException {  4 usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    if (( index != 0 && point.getX() <= points[index - 1].getX()) ||
            (index != pointsCount-1 &&  point.getX() >= points[index + 1].getX()))
        throw new InappropriateFunctionPointException("Неверный X");

    points[index]= new FunctionPoint(point);

}
```

```java
// возвращение координаты x
public double getPointX(int index) throws FunctionPointIndexOutOfBoundsException {  1 usage
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    return points[index].getX();
}
// изменение значения x
public void setPointX(int index, double x) throws InappropriateFunctionPointException {  no usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    if ( ( index != 0 && x <= points[index - 1].getX()) ||
            (index != pointsCount-1 &&  x >= points[index + 1].getX()))
        throw new InappropriateFunctionPointException("Неверный X");

    points[index].setX(x);
}

// возвращения координаты y
public double getPointY (int index) throws FunctionPointIndexOutOfBoundsException {  no usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    return points[index].getY();
}

// изменения значения y
public void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException {  1 usage
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    points[index].setY(y);
}
```

```java
// удаление точки
public void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException {  2 usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    if (pointsCount<3)
        throw new IllegalArgumentException("Недостаточное количество точек");

    System.arraycopy(points, srcPos: index + 1, points, index, length: pointsCount - index - 1);
    points[--pointsCount] = null;

}
```

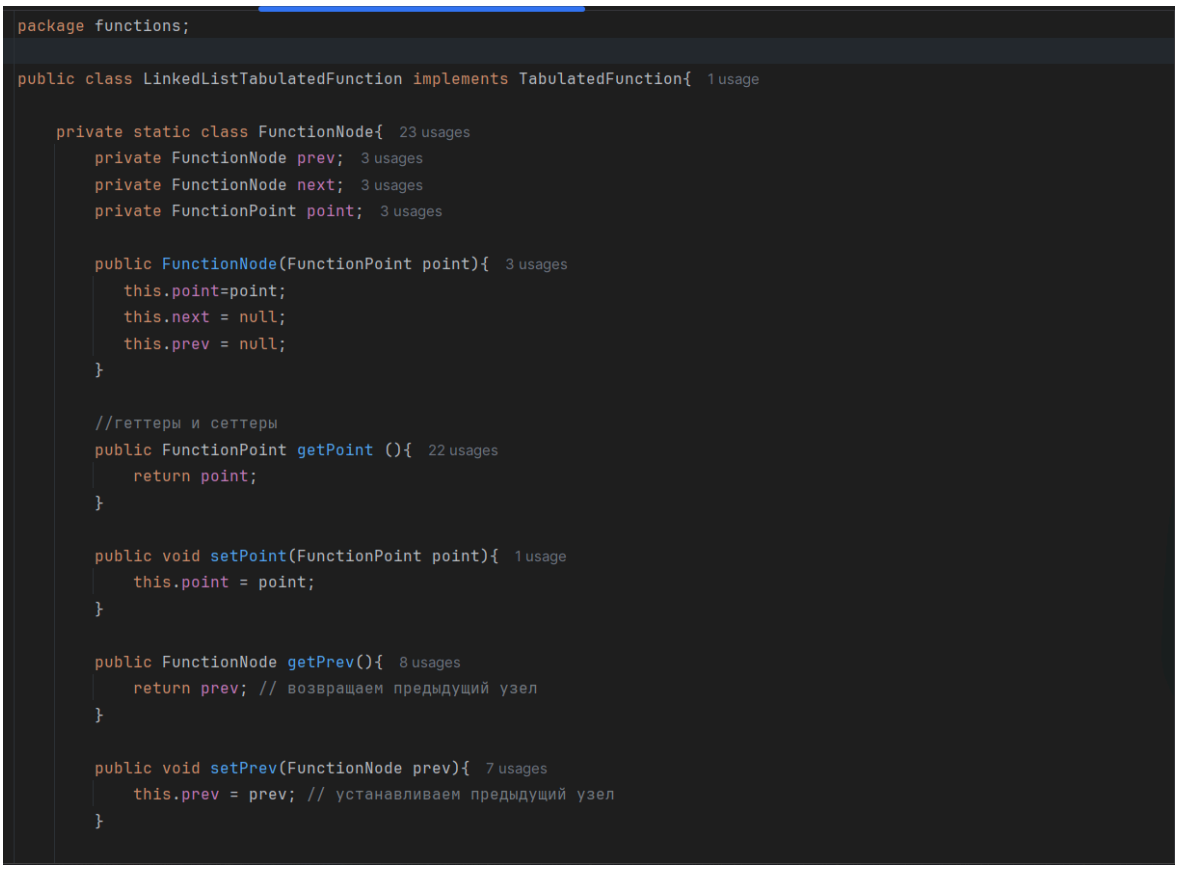
```java
// добавление точки
public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {  1 usage
    int index= 0;

    double newX = point.getX();


    // находим позицию для вставки точки
    while (index < pointsCount && newX > points[index].getX())
        index++;

    if ( Math.abs(points[index].getX() - newX)<1e-10)
        throw new InappropriateFunctionPointException("Точка с таким X уже существует");
    // проверяем нужно ли увеличивать размер массива
    if (pointsCount >= points.length) {
        FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
        System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, pointsCount);
        points = newPoints;
    }

    if (index < pointsCount)
        System.arraycopy(points, index, points, destPos: index + 1, length: pointsCount - index);

    points[index] = new FunctionPoint(point);
    pointsCount++;

}
```

Задание №4

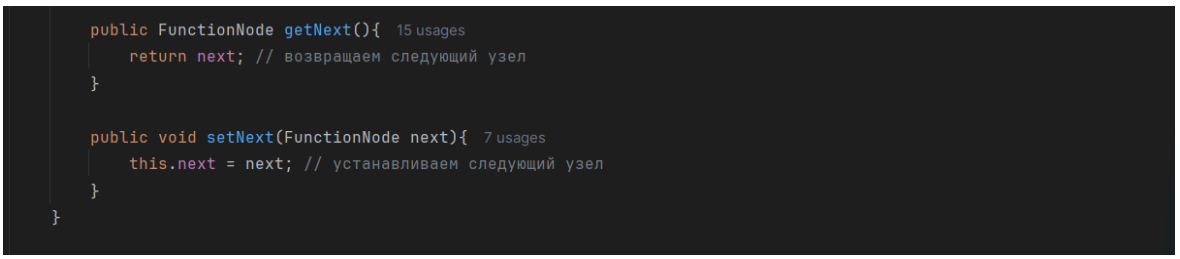
В пакете functions я создала класс LinkedListTabulatedFunction, объект которого также описывает табулированную функцию, но для хранения набора точек, используется двусвязный циклический список. Полями этого класса являются голова списка и количество точек в списке.

```java
private FunctionNode head = new FunctionNode( point: null); // list`s head  23 usages
private int pointsCount;  19 usages
```

Следующим я описала внутренний класс FunctionNode, представляющий узел. Он имеет поля: ссылки на предыдущий и следующий узел и данные о точке; а также геттеры и сеттеры



```
package functions;

public class LinkedListTabulatedFunction implements TabulatedFunction{ 1 usage

    private static class FunctionNode{ 23 usages
        private FunctionNode prev; 3 usages
        private FunctionNode next; 3 usages
        private FunctionPoint point; 3 usages

        public FunctionNode(FunctionPoint point){ 3 usages
            this.point=point;
            this.next = null;
            this.prev = null;
        }

        //геттеры и сеттеры
        public FunctionPoint getPoint (){ 22 usages
            return point;
        }

        public void setPoint(FunctionPoint point){ 1 usage
            this.point = point;
        }

        public FunctionNode getPrev(){ 8 usages
            return prev; // возвращаем предыдущий узел
        }

        public void setPrev(FunctionNode prev){ 7 usages
            this.prev = prev; // устанавливаем предыдущий узел
        }
```



```
        public FunctionNode getNext(){ 15 usages
            return next; // возвращаем следующий узел
        }

        public void setNext(FunctionNode next){ 7 usages
            this.next = next; // устанавливаем следующий узел
        }
    }
```

После я приступила к описанию методов внешнего класса

1) метод FunctionNode getNodeByIndex(int index), возвращающий ссылку на объект элемента списка по его номеру. Для оптимизации доступа к элементам происходит анализ к какой стороне искомый индекс находится ближе: к концу или началу

```
private FunctionNode getNodeByIndex (int index){  13 usages

    FunctionNode curr;
    if (index < pointsCount / 2) {
        // Двигаемся от головы вперед
        curr = head.getNext(); // начинаем с 1ого значащего элемента
        for(int i = 0; i < index; i++) {
            curr = curr.getNext();
        }
    }
    else {
        // Двигаемся от хвоста назад
        curr = head.getPrev(); // начинаем с последнего элемента
        for(int i = pointsCount - 1; i > index; i--) {
            curr = curr.getPrev();
        }
    }

    return curr;
}
```

2) метод FunctionNode addNodeToTail(), добавляющий новый элемент в конец списка и возвращающий ссылку на объект этого элемента. Этот метод реализован обновлением ссылок между узлами, тем самым элемент “привызявается” к текущему последнему элементу списка

```
private FunctionNode addNodeToTail(FunctionPoint point){  3 usages

    FunctionNode tail = head.getPrev();
    FunctionNode newNode = new FunctionNode(point);

    tail.setNext(newNode);
    newNode.setPrev(tail);
    newNode.setNext (head);
    head.setPrev(newNode);

    pointsCount++;

    return newNode;
}
```

3) метод FunctionNode addNodeByIndex(int index), добавляющий новый элемент в указанную позицию списка и возвращающий ссылку на объект этого элемента. Этот метод реализован как и предыдущий: обновлением ссылок между узлами

```
private FunctionNode addNodeByIndex(int index, FunctionPoint point){  1 usage
    FunctionNode newNode = new FunctionNode(point);

    if(index == pointsCount)
        return addNodeToTail(point); // добавляем в конец

    FunctionNode currNode = getNodeByIndex(index);

    newNode.setPrev(currNode.getPrev());
    newNode.setNext (currNode);
    currNode.getPrev().setNext(newNode);
    currNode.setPrev(newNode);

    pointsCount++;

    return newNode;
}
```

4) метод FunctionNode deleteNodeByIndex(int index), удаляющий элемент списка по номеру и возвращающий ссылку на объект удаленного элемента. Он так же реализован как и предыдущие, но элемент списка теперь “отвязывается”

```
private FunctionNode deleteNodeByIndex (int index){  1 usage

    FunctionNode delNode = getNodeByIndex(index);

    delNode.getNext().setPrev(delNode.getPrev());
    delNode.getPrev().setNext(delNode.getNext());

    pointsCount--;

    return delNode;
}
```

Задание №5

Для выполнения этого задания я реализовала те же конструкторы и методы TabulatedFunction но для двусвязного списка

```
// создание табулированной функции
public LinkedListTabulatedFunction(double leftX, double rightX, int pointsCount){  23 usages
    if (leftX>=rightX)
        throw new IllegalArgumentException("Левая граница больше правой");
    if (pointsCount <2)
        throw new IllegalArgumentException("Количество точек меньше двух");

    head.setPrev(head);
    head.setNext(head);
    this.pointsCount = 0;

    double step = Math.abs(rightX-leftX)/(pointsCount-1);
    for (int i=0; i < pointsCount;i++) {
        FunctionPoint point = new FunctionPoint( x: leftX + i*step,  y: 0);
        addNodeToTail(point);
    }
}
```

```java
// создание табулированной функции с заданными значениями по оси ординат
public LinkedListTabulatedFunction(double leftX, double rightX, double[] value){  22 usages
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница больше правой");
    }
    if (value.length < 2) {
        throw new IllegalArgumentException("Количество точек меньше двух");
    }

    head.setPrev(head);
    head.setNext(head);
    this.pointsCount = 0;

    double step = Math.abs(rightX-leftX)/(value.length-1);
    for (int i=0; i < value.length;i++) {
        FunctionPoint point = new FunctionPoint( x: leftX + i*step, value[i]);
        addNodeToTail(point);
    }
}
```

для получения левой и правой границы области определения обращаемся к первому значащему элементу списку (следующий после головы) и последнему (элемент до головы)

```java
// возвращение левой границы области определения
public double getLeftDomainBorder() { return head.getNext().getPoint().getX(); }

// возвращение правой границы области определения
public double getRightDomainBorder() { return head.getPrev().getPoint().getX(); }
```

метод getFunctionalValue я оптимизировала прямым обходом списка вместо обращения к каждому элементу по индексу

```java
// возвращение y, если точка лежит в области определения функции
public double getFunctionValue(double x){  1 usage
    double leftX = getLeftDomainBorder();
    double rightX = getRightDomainBorder();

    if (x>= leftX && x <= rightX){

        FunctionNode curr = head.getNext();
        while (curr != head && curr.getNext() != head) {
            double x1 = curr.getPoint().getX();
            double x2 = curr.getNext().getPoint().getX();

            // совпадает ли x
            if (Math.abs(x - x1) < 1e-10)
                return curr.getPoint().getY();
            if (Math.abs(x - x2) < 1e-10)
                return curr.getNext().getPoint().getY();
            // попадает ли x в интервал
            if (x > x1 && x < x2) {
                double y1 = curr.getPoint().getY();
                double y2 = curr.getNext().getPoint().getY();
                return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
            }
            // переходим к следующему
            curr = curr.getNext();
        }
    }
    return Double.NaN;
}
```

```java
// возвращения количества точек
public int getPointsCount() { return pointsCount; }

// возвращение точки
public FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException {  1 usage
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    double x = getNodeByIndex(index).getPoint().getX();
    double y = getNodeByIndex(index).getPoint().getY();
    return new FunctionPoint(x, y);
}

// изменение указанной точки на заданную
public void setPoint (int index, FunctionPoint point) throws InappropriateFunctionPointException {  2 usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    if (( index != 0 && point.getX() <= getNodeByIndex(index-1).getPoint().getX()) ||
            (index != pointsCount-1 &&  point.getX() >= getNodeByIndex(index+1).getPoint().getX()))
        throw new InappropriateFunctionPointException("Неверный X");

    getNodeByIndex(index).setPoint(new FunctionPoint(point));

}
```

```java
// возвращение координаты x
public double getPointX(int index) throws FunctionPointIndexOutOfBoundsException {  1 usage
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    return getNodeByIndex(index).getPoint().getX();
}
// изменение значения x
public void setPointX(int index, double x) throws InappropriateFunctionPointException {  no usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    if ( ( index != 0 && x <= getNodeByIndex(index-1).getPoint().getX()) ||
            (index != pointsCount-1 &&  x >= getNodeByIndex(index+1).getPoint().getX()))
        throw new InappropriateFunctionPointException("Неверный X");

    getNodeByIndex(index).getPoint().setX(x);
}

// возвращения координаты y
public double getPointY (int index) throws FunctionPointIndexOutOfBoundsException {  no usages
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    return getNodeByIndex(index).getPoint().getY();
}

// изменения значения y
public void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException {  1 usage
    if (index< 0 || index >= pointsCount)
        throw new FunctionPointIndexOutOfBoundsException("Не существует точки");
    getNodeByIndex(index).getPoint().setY(y);
}
```

как и метод getFunctionalValue я оптимизировала методы addPoint и вспомогательный для вывода outFunction: обход списка производится

прямым обращением к элементам, а не обращением по индексу

```java
// добавление точки
public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {  1 usage
    int index= 0;

    double newX = point.getX();

    // находим позицию для вставки точки
    FunctionNode curr = head.getNext();
    while (curr != head && newX > curr.getPoint().getX()) {
        curr = curr.getNext();
        index++;
    }

    if (curr != head && Math.abs(curr.getPoint().getX() - newX) < 1e-10)
        throw new InappropriateFunctionPointException("Точка с таким X уже существует");

    addNodeByIndex(index, point);

}

// вывод массива точек
public void outFunction(){  4 usages
    FunctionNode curr = head.getNext();
    while (curr != head) {
        System.out.printf("(%.2f, %.2f) ", curr.getPoint().getX(), curr.getPoint().getY());
        curr = curr.getNext();
    }
    System.out.println();
}
```

Задание №6

я переименовала класс TabulatedFunction в класс ArrayTabulatedFunction и создать интерфейс TabulatedFunction, содержащий объявления общих методов классов ArrayTabulatedFunction и LinkedListTabulatedFunction, в класс прописала ключевое слово implements TabulatedFunction

```java
package functions;

public interface TabulatedFunction{// количество точек  5 usages  2 implementations

    // возвращение левой границы области определения
    public double getLeftDomainBorder();  2 usages  2 implementations

    // возвращение правой границы области определения
    public double getRightDomainBorder();  2 usages  2 implementations

    // возвращение y, если точка лежит в области определения функции
    public double getFunctionValue(double x);  1 usage  2 implementations

    // возвращения количества точек
    public int getPointsCount();  1 usage  2 implementations

    // возвращение точки
    public FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException ;  1 usage  2 implementations

    // изменение указанной точки на заданную
    public void setPoint (int index, FunctionPoint point) throws InappropriateFunctionPointException ;  2 usages  2 implemen

    // возвращение координаты x
    public double getPointX(int index) throws FunctionPointIndexOutOfBoundsException ;  1 usage  2 implementations

    // изменение значения x
    public void setPointX(int index, double x) throws InappropriateFunctionPointException ;  no usages  2 implementations

    // возвращения координаты y
    public double getPointY (int index) throws FunctionPointIndexOutOfBoundsException;  no usages  2 implementations
```

```java
    // возвращения координаты y
    public double getPointY (int index) throws FunctionPointIndexOutOfBoundsException;  no usages  2 implementations

    // изменения значения y
    public void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException ;  1 usage  2 implementations

    // удаление точки
    public void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException ;  4 usages  2 implementations

    // добавление точки
    public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  1 usage  2 implementations

    // вывод массива точек
    public void outFunction();  4 usages  2 implementations
}
```

Задание №7

это задание заключается в тестировании написанного кода. Для проверки выполнения методов с валидными данными я использовала двусвязный список, чтобы проверить правильность его написания. Для тестирования

выброса исключений я использовала заведомо некорректные данные, в результате чего программа повела себя ожидаемо и поймала все ошибки

```
PS C:\Users\nastya\JavaProjects\Lab-3-2025> javac Main.java
PS C:\Users\nastya\JavaProjects\Lab-3-2025> java Main
==Тестирование двусвязного списка==
Функция: y=1.2x
(1,00, 1,20) (1,66, 1,99) (2,31, 2,78) (2,97, 3,57) (3,63, 4,35) (4,29, 5,14) (4,94, 5,93) (5,60, 6,72)
Проверка удаления точки с индексом 3
(1,00, 1,20) (1,66, 1,99) (2,31, 2,78) (3,63, 4,35) (4,29, 5,14) (4,94, 5,93) (5,60, 6,72)
Проверка добавления точки (3, 4)
(1,00, 1,20) (1,66, 1,99) (2,31, 2,78) (3,00, 4,00) (3,63, 4,35) (4,29, 5,14) (4,94, 5,93) (5,60, 6,72)
Проверка установки точки (2.2, 5) на позицию с индексом 2
(1,00, 1,20) (1,66, 1,99) (2,20, 5,00) (3,00, 4,00) (3,63, 4,35) (4,29, 5,14) (4,94, 5,93) (5,60, 6,72)
Проверка getFunctionValue
(-1,00, NaN)
(-0,20, NaN)
(0,60, NaN)
(1,40, 1,68)
(2,20, 5,00)
==Тестирование исключений==
Некорректное создание (левая граница больше правой)
Error Левая граница больше правой
обращение к точке с несуществующим индексом
Error Не существует точки
точка нарушает порядок возрастания
Error Неверный X
Error Недостаточное количество точек
PS C:\Users\nastya\JavaProjects\Lab-3-2025>
```

```
System.out.println("==Тестирование исключений==");

System.out.println("Некорректное создание (левая граница больше правой)");
try {
    TabulatedFunction testFunc = new ArrayTabulatedFunction( leftX: 34,  rightX: 3,  pointsCount: 8);
    System.out.println("без ошибок");
}catch (IllegalArgumentException e){
    System.out.println("Error " + e.getMessage());
}

System.out.println("обращение к точке с несуществующим индексом");
try {
    func.getPoint( index: 100);
    System.out.println("Без ошибок");
}catch (FunctionPointIndexOutOfBoundsException e){
    System.out.println("Error " + e.getMessage());
}

System.out.println("точка нарушает порядок возрастания");
try {
    func.setPoint( index: 2, new FunctionPoint( x: 30, y: 2));
    System.out.println("без ошибок");
}catch (InappropriateFunctionPointException e){
    System.out.println("Error " + e.getMessage());
}
```

```
try{
    TabulatedFunction testFunc2 = new ArrayTabulatedFunction( leftX: 0, rightX: 2, pointsCount: 3);
    testFunc2.deletePoint( index: 0);
    testFunc2.deletePoint( index: 0);
    testFunc2.deletePoint( index: 0);
    System.out.println("без ошибок");
}catch (IllegalStateException e){
    System.out.println("Error " + e.getMessage());
}
```