ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №2

Студент Кольчугина Е. А.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Содержание

Задание 1

Создан пакет functions для организации классов программы. Все последующие классы размещены в данном пакете.

Задание 2

Users > lisako > Documents > oop > Lab-2-2025 > functions > J FunctionPoint.java

```java
package functions;

public class FunctionPoint{
    private double x;
    private double y;

    public double getX(){
        return x;
    }
    public void setX(double x){
        this.x = x;
    }

    public double getY(){
        return y;
    }
    public void setY(double y){
        this.y = y;
    }


    public FunctionPoint(double x, double y){
        this.x = x;
        this.y = y;
    }

    public FunctionPoint(FunctionPoint point){
        this.x = point.x;
        this.y = point.y;
    }

    public FunctionPoint(){
        this.x = 0;
        this.y = 0;
    }
}
```

Код 1: FunctionPoint.java

Реализован публичный класс FunctionPoint, входящий в пакет functions. Для учета особенностей инкапсуляции переменные x и y объявлены приватными, доступ к ним возможен только через публичные методы. Добавлены методы getX и getY для получения значений переменных и методы setX и setY для установки значений. Созданы три конструктора:

- FunctionPoint(double x, double y) - создает точку с заданными координатами
- FunctionPoint(FunctionPoint point) - создает копию существующей точки
- FunctionPoint() - создает точку (0, 0)

Задание 3

C: > Users > lisako > Documents > oop > Lab-2-2025 > functions > TabulatedFunction.java

```java
package functions;

public class TabulatedFunction{
    private FunctionPoint[] points;
    private int pointsCount;

    public TabulatedFunction(double leftX, double rightX, int pointsCount){
        points = new FunctionPoint[pointsCount + 2];
        this.pointsCount = pointsCount;
        double distance = (rightX - leftX)/(pointsCount - 1);

        for (int i = 0; i < pointsCount; i++){
            double x = leftX + i * distance;
            points[i] = new FunctionPoint(x, 0.0);
        }
    }

    public TabulatedFunction(double leftX, double rightX, double[] values){
        this.pointsCount = values.length;
        points = new FunctionPoint[pointsCount + 2];
        double distance = (rightX - leftX)/(pointsCount - 1);

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

Код 2: TabulatedFunction.java

Реализован публичный классTabulatedFunction, входящий в пакет functions. Объявлены привптные поля:

- points - массив для хранения точек функции
- pointsCount - фактическое количество точек

Реализованы конструкторы:

- TabulatedFunction(leftX, rightX, pointsCount) - создание табулированной функции с равномерным распределением точек, y=0:

В качестве параметров передаем левую и правую границы интервала для x и количество точек. Создаем пустой массив с запасом памяти, вычисляем шаг между точками и создаем упорядоченный по возрастанию массив точек с равными интервалами для x.

- TabulatedFunction(leftX, rightX, values) - создание табулированной функции с заданными значениями y:

  В качестве параметров передаем левую и правую границы интервала для x и массив значений функции. Устанавливаем количество точек, равное длине массива значений, создаем пустой массив с запасом памяти, вычисляем шаг между точками и создаем упорядоченный по возрастанию массив точек с равными интервалами для x и с значениями y из массива values.

Задание 4

C: > Users > lisako > Documents > oop > Lab-2-2025 > functions > J TabulatedFunction.java

```
3   public class TabulatedFunction{
29      public double getLeftDomainBorder(){
30          return points[0].getX();
31      }
32
33      public double getRightDomainBorder() {
34          return points[pointsCount - 1].getX();
35      }
36
37      public double getFunctionValue(double x){
38          if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
39              return Double.NaN;
40          }
41          for (int i = 0; i < pointsCount - 1; i++){
42              double x_1 = points[i].getX();
43              double x_2 = points[i + 1].getX();
44
45              if (x == x_1){
46                  return points[i].getY();
47              }
48
49              if (x == x_2){
50                  return points[i + 1].getY();
51              }
52
53              if (x > x_1 && x < x_2){
54                  double y_1 = points[i].getY();
55                  double y_2 = points[i + 1].getY();
56
57                  return (x - x_1) * (y_2 - y_1) / (x_2 - x_1) + y_1;
58              }
59          }
60          return Double.NaN;
61      }
```

Код 3: TabulatedFunction.java

Реализованы методы:

- getLeftDomainBorder() - для получения левой границы области определения функции:
  Возвращает координату x первой точки массива
- getRightDomainBorder() - для получения правой границы области определения функции:
  Возвращает координату x последней точки массива
- getFunctionValue(double x) - вычисляет значение функции в точке x с использованием линейной интерполяции:
  При прохождении точкой проверки на принадлежность интервалу запускается цикл: для каждого интервала проверяется совпадает ли x с одной из границ и, если нет, применяется формула линейной интерполяции: $\frac{y-y_1}{y_2-y_1} = \frac{x-x_1}{x_2-x_1}$

Задание 5

C: > Users > lisako > Documents > oop > Lab-2-2025 > functions > J TabulatedFunction.java

```
3   public class TabulatedFunction{
62
63      public int getPointsCount(){
64          return pointsCount;
65      }
66
67      public FunctionPoint getPoint(int index){
68          return new FunctionPoint(points[index]);
69      }
70
71      public void setPoint(int index, FunctionPoint point){
72          if (index < 0 || index >= pointsCount) {
73              return;
74          }
75          if (index > 0 && point.getX() <= points[index - 1].getX()) {
76              return;
77          }
78          if (index < pointsCount - 1 && point.getX() >= points[index + 1].getX()) {
79              return;
80          }
81
82          points[index] = new FunctionPoint(point);
83      }
84
85      public double getPointX(int index){
86          return points[index].getX();
87      }
88
89      public void setPointX(int index, double x){
90          if (index > 0 && x <= points[index - 1].getX()) {
91              return;
92          }
93          if (index < pointsCount - 1 && x >= points[index + 1].getX()) {
94              return;
95          }
96
97          points[index].setX(x);
98      }
99
100     public double getPointY(int index){
101         return points[index].getY();
102     }
103
104      public void setPointY(int index, double y) {
105         points[index].setY(y);
106     }
```

Код 4: TabulatedFunction.java

Реализованы методы:

- int getPointsCount() - возвращает количество точек
- FunctionPoint getPoint(int index) - возвращает точку по индексу:
  Используя конструктор копирования, возвращает копию точки, соответствующей индексу.
- void setPoint(int index, FunctionPoint point) - заменяет точку с проверкой упорядоченности:
  Проверяет соответствие индекса интервалу и порядку слева и справа и заменяет выбранную точку созданной копией.
- double getPointX(int index) - получает x-координату точки (используя метод getX)
- void setPointX(int index, double x) - изменяет x-координату (после проверки на порядок использует метод setX)
- double getPointY(int index) - получает y-координату точки (используя метод getY)
- void setPointY(int index, double y) - изменяет y-координату

Задание 6

C: > Users > lisako > Documents > oop > Lab-2-2025 > functions > J TabulatedFunction.java

```
3   public class TabulatedFunction{
111     public void deletePoint(int index) {
112         if (pointsCount <= 2) {
113             return;
114         }
115         if (index < 0 || index >= pointsCount) {
116             return;
117         }
118
119         System.arraycopy(points, index + 1, points, index, pointsCount - index - 1);
120         pointsCount--;
121         points[pointsCount] = null;
122     }
123
124     public void addPoint(FunctionPoint point) {
125         FunctionPoint newPoint = new FunctionPoint(point);
126         int newIndex = 0;
127         while (newIndex < pointsCount && points[newIndex].getX() < newPoint.getX()) {
128          newIndex++;
129         }
130         if  (newIndex < pointsCount && points[newIndex].getX() == newPoint.getX()) {
131             return;
132         }
133         if (pointsCount == points.length) {
134             increaseArraySize();
135         }
136
137         System.arraycopy(points, newIndex, points, newIndex + 1, pointsCount - newIndex);
138         points[newIndex] = newPoint;
139         pointsCount++;
140     }
141
142     private void increaseArraySize() {
143         FunctionPoint[] newArray = new FunctionPoint[points.length * 2 + 2];
144         System.arraycopy(points, 0, newArray, 0, pointsCount);
145         points = newArray;
146     }
147 }
```

Код 5: TabulatedFunction.java

Реализованы методы:

- void deletePoint(int index) - удаляет точку по указанному индексу:

Для табулированной функции необходимо наличие хотя бы двух точек, поэтому удаление не произойдет, если число точек меньше или равно 2.

Метод проверяет соответствует ли индекс интервалу. Используя метод arraycopy() класса System, копирует массив после выбранного элемента и сдвигает его на 1 элемент влево. Затем уменьшает число точек на один и удаляет последнюю точку.

Таким образом, массив не пересоздается, а элементы сдвигаются внутри существующего.

- void addPoint(FunctionPoint point) - добавляет новую точку, сохраняя упорядоченность:
  Создает копию переделанной точки и, используя цикл while, ищет значение индекса для новой точки (в каждом цикле прибавляет 1 к индексу, пока не найдет ближайший x справа к новой точке). При совпадении количества точек с длиной массива применяет метод increaseArraySize() для увеличения массива.
  Используя метод arraycopy() класса System, копирует все элементы после (включая элемент, на место которого встанет новая точка) и перемещает их вправо на 1. Увеличивает счетчик количества точек на 1
- increaseArraySize() - приватный метод, необходимый для увеличения числа элементов массива при совпадении его длины с количеством точек в нем (используется только при добавлении новой точки):
  Создает новый массив большего размера, копирует в него данные предыдущего и заменяет его имя на имя предыдущего.

Задание 7

C: > Users > lisako > Documents > oop > Lab-2-2025 > J Main.java

```java
import functions.TabulatedFunction;
import functions.FunctionPoint;

public class Main {
    public static void main(String[] args) {
        // y = x^3, x = [0, 5]
        double[] values = {0, 1, 8, 27, 64, 125};
        TabulatedFunction func = new TabulatedFunction(0, 5, values);

        System.out.println("Function^ y = x^3, x = [0, 5]");
        printFunctionInfo(func);

        System.out.println("\ncalculating values");
        double[] testPoints = {-1, 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 5};
        for (double x : testPoints) {
            double result = func.getFunctionValue(x);
            if (Double.isNaN(result)) {
                System.out.printf("f(%.1f) = not defined%n", x);
            } else {
                System.out.printf("f(%.1f) = %.2f%n", x, result);
            }
        }

        System.out.println("\nadding points");
        func.addPoint(new FunctionPoint(0.5, 0.125));
        func.addPoint(new FunctionPoint(4, 64));
        func.addPoint(new FunctionPoint(2.5, 15.625));
        System.out.println("result:");
        printFunctionInfo(func);

        System.out.println("\ncheck");
        for (double x : new double[]{0.5, 2.5, 4}) {
            System.out.printf("f(%.1f) = %.3f%n", x, func.getFunctionValue(x));
        }

        System.out.println("\npoint changing");
        func.setPointY(2, 5.0);
        System.out.println("result:");
        printFunctionInfo(func);
        System.out.printf("f(%.1f) = %.3f%n", func.getPointX(2), func.getFunctionValue(func.getPointX(2)));

        System.out.println("\npoint deleting");
        func.deletePoint(3);
        System.out.println("result:");
        printFunctionInfo(func);

    }

    private static void printFunctionInfo(TabulatedFunction func) {
        System.out.println("Область определения: [" + func.getLeftDomainBorder() + ", " + func.getRightDomainBorder() + "]");
        System.out.println("Количество точек: " + func.getPointsCount());
        System.out.print("Точки: ");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.print("(" + func.getPointX(i) + "; " + func.getPointY(i) + ")");
            if (i < func.getPointsCount() - 1) {
                System.out.print(", ");
            }
        }
        System.out.println();
    }
}
```

Код 5: Main.java

Реализован класс Main для проверки работы написанных классов:

- Создает табулированную функцию, используя массив значений values
- Тестирует вычисление значений (массив из x для проверки: значения для x не из интервала не определены; для x, совпадающих с узлами, значения соответствуют заданным; для x из интервалов между 2 узлами значения функции вычислены с применением линейной интерполяции)
- Тестирует добавление точек (создает новый объект с заданными координатами с помощью конструктора и вызывает метод addPoint() объекта func)
- Тестирует изменение точек (при помощи метода setPointY для данного x меняется значение функции)
- Тестирует удаление точки (при помощи метода deletePoint(index)

В класс также добавлен вспомогательный приватный метод printFunctionInfo:

- Он является статическим (принадлежит классу main и является единственным для всех объектов)
- Выводит область определения, количество точек и сами точки

Консоль:

lisako@lisa MINGW64 ~/Documents/oop/Lab-2-2025 (main)

$ javac functions/*.java

$ javac Main.java

$ java Main

Function^ y = x^3, x = [0, 5]

Область определения: [0.0, 5.0]

Количество точек: 6

Точки: (0.0; 0.0), (1.0; 1.0), (2.0; 8.0), (3.0; 27.0), (4.0; 64.0), (5.0; 125.0)

calculating values

f(-1,0) = not defined

f(0,0) = 0,00

f(0,5) = 0,50

f(1,0) = 1,00

f(1,5) = 4,50

f(2,0) = 8,00

f(2,5) = 17,50

f(3,0) = 27,00

f(3,5) = 45,50

f(4,0) = 64,00

f(5,0) = 125,00

adding points

result:

Область определения: [0.0, 5.0]

Количество точек: 8

Точки: (0.0; 0.0), (0.5; 0.125), (1.0; 1.0), (2.0; 8.0), (2.5; 15.625), (3.0; 27.0), (4.0; 64.0), (5.0; 125.0)

check

f(0,5) = 0,125

f(2,5) = 15,625

f(4,0) = 64,000

point changing

result:

Область определения: [0.0, 5.0]

Количество точек: 8

Точки: (0.0; 0.0), (0.5; 0.125), (1.0; 5.0), (2.0; 8.0), (2.5; 15.625), (3.0; 27.0), (4.0; 64.0), (5.0; 125.0)

f(1,0) = 5,000

point deleting

result:

Область определения: [0.0, 5.0]

Количество точек: 7

Точки: (0.0; 0.0), (0.5; 0.125), (1.0; 5.0), (2.5; 15.625), (3.0; 27.0), (4.0; 64.0), (5.0; 125.0)