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 Ганжа Дарья

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Task 1

Я создала пакет functions, в котором далее будут создаваться классы программы.

```
package functions;
```

Task 2

В пакете functions создала класс FunctionPoint, объект которого должен описывает точку табулированной функции.

В классе описаны следующие конструкторы:

• FunctionPoint(double x, double y) – создаёт объект точки с заданными координатами;

• FunctionPoint(FunctionPoint point) – создаёт объект точки с теми же координатами, что у указанной точки;

• FunctionPoint() – создаёт точку с координатами (0; 0).

Также добавлены геттеры и сеттеры для получения значений x и y

```
package functions;
public class FunctionPoint{
    private double x;
    private double y;

   public FunctionPoint(double x, double y){
    this.x=x;
    this.y=y;
   }
   public FunctionPoint(FunctionPoint point){
    this(point.x, point.y);
   }
   public FunctionPoint(){
    this.x=0.0;
    this.y=0.0;
   }
   public double getX(){
    return x;
   }
    public double getY(){
    return y;
   }
   public void setX(double x){
    this.x=x;
   }
    public void setY(double y){
    this.y=y;
   }
}
```

Task 3

В пакете functions создан класс TabulatedFunction, объект которого описывает табулированную функцию.

Для хранения данных о точках используется массив типа FunctionPoint.

В классе описаны следующие конструкторы:

• TabulatedFunction(double leftX, double rightX, int pointsCount) – создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования (значения функции в точках при этом следует считать равными 0);

```
public TabulatedFunction(double leftX, double rightX, int pointsCount) {
    this.points = new FunctionPoint[pointsCount];

    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + step * i;
        points[i] = new FunctionPoint(x, 0.0);
    }
}
```

• TabulatedFunction(double leftX, double rightX, double[] values) – аналогичен предыдущему конструктору, но вместо количества точек получает значения функции в виде массива.

```
public TabulatedFunction(double leftX, double rightX, double[] values) {
    this.points = new FunctionPoint[values.length];

    double step = (rightX - leftX) / (values.length - 1);
    for (int i = 0; i < values.length; i++) {
        double x = leftX + step * i;
        points[i] = new FunctionPoint(x, values[i]);
    }
}
```

В обоих случаях точки создаются через равные интервалы по x.

Задание 4

В класс TabulatedFunction также добавлены следующие методы:

• Метод double getLeftDomainBorder() возвращает значение левой границы области определения табулированной функции.

```
public double getLeftDomainBorder() {
    return points[0].getX();
}
```

• Метод double getRightDomainBorder() возвращает значение правой границы области определения табулированной функции.

```
public double getRightDomainBorder() {
    return points[points.length - 1].getX();
}
```

• Метод double getFunctionValue(double x) возвращает значение функции в точке x, если эта точка лежит в области определения функции. В противном случае метод должен возвращает значение неопределённости. При расчёте значения функции следует использовалась линейная интерполяция.

```
public double getFunctionValue(double x) {
 double leftBorder = getLeftDomainBorder();
 double rightBorder = getRightDomainBorder();

 if (x < leftBorder || x > rightBorder) {
    return Double.NaN;
 }


 if (x == leftBorder) {
    return points[0].getY();
 }
 if (x == rightBorder) {
    return points[points.length - 1].getY();
 }


 for (int i = 0; i < points.length - 1; i++) {
    double x1 = points[i].getX();
    double x2 = points[i + 1].getX();


    if (x >= x1 && x <= x2) {
        double y1 = points[i].getY();
        double y2 = points[i + 1].getY();


        return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
    }
 }
```

Задание 5

В классе TabulatedFunction также описаны методы, необходимые для работы с точками табулированной функции.

• Метод int getPointsCount() возвращает количество точек.

```
public int getPointsCount() {
    return points.length;
}
```

• Метод FunctionPoint getPoint(int index) возвращает копию точки, соответствующей переданному индексу.

```
public FunctionPoint getPoint(int index) {
    return new FunctionPoint(points[index]);
}
```

• Метод void setPoint(int index, FunctionPoint point) заменяет указанную точку табулированной функции на переданную.

```
public void setPoint(int index, FunctionPoint point) {
    double newX = point.getX();

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

• Метод double getPointX(int index) возвращает значение абсциссы точки с указанным номером.

```
public double getPointX(int index) {
    return points[index].getX();
}
```

• Метод void setPointX(int index, double x) изменяет значение абсциссы точки с указанным номером.

```
public void setPointX(int index, double x) {
    double currentY = points[index].getY();
    points[index] = new FunctionPoint(x, currentY);
}
```

• Метод double getPointY(int index) возвращает значение ординаты точки с указанным номером.

```
public double getPointY(int index) {
    return points[index].getY();
}
```

• Метод void setPointY(int index, double y) изменяет значение ординаты точки с указанным номером.

```
public void setPointY(int index, double y) {
    points[index].setY(y);
}
```

Задание 6

В классе TabulatedFunction описаны методы, изменяющие количество точек табулированной функции.

• Метод void deletePoint(int index) удаляет заданную точку табулированной функции.

```
public void deletePoint(int index) {
    FunctionPoint[] newPoints = new FunctionPoint[points.length - 1];
    System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, index);
    System.arraycopy(points, index + 1, newPoints, index, points.length - index - 1);
    points = newPoints;
}
```

• Метод void addPoint(FunctionPoint point) добавляет новую точку табулированной функции.

```
public void addPoint(FunctionPoint point) {

    int insertIndex = 0;
    while (insertIndex < points.length && points[insertIndex].getX() < point.getX()) {
        insertIndex++;
    }
   FunctionPoint[] newPoints = new FunctionPoint[points.length + 1];
    System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, insertIndex);
    newPoints[insertIndex] = new FunctionPoint(point);
    System.arraycopy(points, insertIndex, newPoints, insertIndex + 1, points.length - insertIndex);
    points = newPoints;
}
```

Задание 7

класс Main, содержащий точку входа программы.

В методе main() присутствует экземпляр класса TabulatedFunction и задайте для него табулированные значения какой-нибудь известной вам функции.

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

public class Main {
    Run | Debug
    public static void main(String[] args){
        double[] values = {0.0, 1.0, 4.0, 16.0};
        TabulatedFunction function = new TabulatedFunction(leftX: 0.0,rightX: 4.0, values);
        printFunctionInfo(function);

        System.out.println(x: "\nFunction's values:");
        double[] points={-1.0, 0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0};
        for (double x: points){
            double y = function.getFunctionValue(x);
            System.out.printf(format: "f(%.1f) = %s%n", x, Double.isNaN(y) ? "undefined" : String.format(format: "%.2f", y));
        }
        System.out.println(x: "\n Changing points");
        function.setPointY(index: 2,y: 5.0);
        System.out.println(x: "After changing points");
        printFunctionInfo(function);

        System.out.println(x: "Adding points");
        FunctionPoint newPoint = new FunctionPoint(1.5,2.5);
        function.addPoint(newPoint);
        System.out.println(x: "After adding");
        printFunctionInfo(function);

        System.out.println(x: "Deleting points:");
        function.deletePoint(index: 2);
        System.out.println(x: "After deleting points:");
        printFunctionInfo(function);

        System.out.println(x: "Deleting points:");
        function.deletePoint(index: 2);
        System.out.println(x: "After deleting points:");
        printFunctionInfo(function);

        System.out.println(x: "\n New Function's values:");
        for (double x: points){
            double y = function.getFunctionValue(x);
            System.out.printf(format: "f(%.1f) = %s%n", x, Double.isNaN(y) ? "undefined" : String.format(format: "%.2f", y));
        }
        System.out.println(x: "\nBorders:");
        System.out.println("\n Left border: " + function.getLeftDomainBorder());
        System.out.println("\n Right border: " + function.getRightDomainBorder());
        System.out.println("\n Counts of points: " + function.getPointsCount());
    }
    private static void printFunctionInfo(TabulatedFunction function) {
        System.out.println(x: "Function's points:");
        for (int i = 0; i < function.getPointsCount(); i++) {
            double x = function.getPointX(i);
            double y = function.getPointY(i);
            System.out.printf(format: " %d %.1f %.1f %n", i, x, y);
        }
```

Результат:

Function's points:

0 0,0 0,0

1 1,3 1,0

2 2,7 4,0

3 4,0 16,0

Function's values:

f(-1,0) = undefined

f(0,0) = 0,00

f(0,5) = 0,38

f(1,0) = 0,75

f(1,5) = 1,38

f(2,0) = 2,50

f(2,5) = 3,63

f(3,0) = 7,00

f(3,5) = 11,50

f(4,0) = 16,00

f(5,0) = undefined

Changing points

After changing points

Function's points:

0 0,0 0,0

1 1,3 1,0

2 2,7 5,0

3 4,0 16,0

Adding points

After adding

Function's points:

0 0,0 0,0

1 1,3 1,0

2 1,5 2,5

3 2,7 5,0

4 4,0 16,0

Deleting points:

After deleting points:

Function's points:

0 0,0 0,0

1 1,3 1,0

2 2,7 5,0

3 4,0 16,0

New Function's values:

f(-1,0) = undefined

f(0,0) = 0,00

f(0,5) = 0,38

f(1,0) = 0,75

f(1,5) = 1,50

f(2,0) = 3,00

f(2,5) = 4,50

f(3,0) = 7,75

f(3,5) = 11,88

f(4,0) = 16,00

f(5,0) = undefined

Borders:

Left border: 0.0

Right border: 4.0

Counts of points: 4