Лабораторная работа №2

Валиневич Владислав Александрович
группа: 6204-010302D

Цель лабораторной работы: *Разработать набор классов для работы с функциями одной переменной, заданными в табличной форме.*

Ход выполнения:

Задание 1:

Создадим пакет *functions,* далее в нем будем хранить классы программы.

Задание 2:

Создадим класс *FunctionPoint*, который реализован для описания точки табулированной функции с полями x и y.
Также реализуем конструкторы *FunctionPoint(double x, double y)*, *FunctionPoint(FunctionPoint point)* и *FunctionPoint()*.
Далее надо понадобятся геттеры и сеттеры

```
package functions;

public class FunctionPoint {
    double x;
    double y;


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


    public FunctionPoint(FunctionPoint p) {
        x = p.x;
        y = p.y;
    }


    public FunctionPoint() {
        x = 0;
        y = 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;
    }

}
```

Задание 3:

В том же пакете *functions* создадим новый класс *TabulatedFunction*
Класс реализует табулированную функцию, хранящую точки в массиве *FunctionPoint[] points*, упорядоченном по возрастанию X.
Опишем конструкторы *TabulatedFunction(double leftX, double rightX, int pointsCount)* и *TabulatedFunction(double leftX, double rightX, double[] values).*
Конструктор 1 - заполняет массив точками с равномерным шагом
Конструктор 2 - заполняет массив точками с заданными значениями y.

```
package functions;


public class TabulatedFunction {
    private FunctionPoint[] points; //массив точек

    // Конструктор 1:
    public TabulatedFunction(double leftX, double rightX, int pointsCount) {

     this.points = new FunctionPoint[pointsCount];
     double step = (rightX - leftX) / (pointsCount - 1); //вычисляем шаг между точками

     // Заполняем массив точками с координатой x и с y=0
     for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, 0);
        }
    }

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

     //заполняем массив заданным значением y
     for (int i = 0; i < values.length; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, values[i]);
        }
    }
```

Задание 4:

Опишем методы для работы с функцией:

double getLeftDomainBorder() - возвращаем координату x первой точки.

double getRightDomainBorder() - возвращает координату x последней точки.

double getFunctionValue(double x) – вычисляет значение функции в точке x и проверяет входит ли в область определения.

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

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

//Вычисляем значение функции
  public double getFunctionValue(double x) {

        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }


        for (int i = 0; i < points.length - 1; i++) {
            double x1 = points[i].getX();
            double x2 = points[i + 1].getX();
            double y1 = points[i].getY();
            double y2 = points[i + 1].getY();


            if (x == x1) {
                return y1;
            }
            if (x == x2) {
                return y2;
            }


            if (x > x1 && x < x2) {
                return y1 + (y2 - y1) / (x2 - x1) * (x - x1);
            }
        }


        return points[points.length - 1].getY();
    }
```

Алгоритм работает так, что для заданного x находится интервал между двумя точками, значение вычисляется по формуле прямой через две точки.

Задание 5:
Методы работы с точками.
getPointsCount()– возвращает количество точек.
getPoint(int index)– возвращает копию точки .
setPoint(int index, FunctionPoint point)– заменяет точку.
getPointX(int index), setPointX(int index, double x) – работа с x.
getPointY(int index), setPointY(int index, double y) – работа с y.

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

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

//Копируем точку
  public void setPoint(int index, FunctionPoint point){

        if (index > 0 && point.getX() <= points[index - 1].getX()){
            return;
        }
        if (index < points.length - 1 && point.getX() >= points[index + 1].getX()){
            return;
        }
        points[index] = new FunctionPoint(point);
    }

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

  public void setPointX(int index, double x){
        if (index > 0 && x <= points[index - 1].getX()){
            return;
        }
        if (index < points.length - 1 && x >= points[index + 1].getX()){
            return;
        }
        points[index].setX(x);
    }

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


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

Задание 6:
deletePoint(int index) - удаляет заданную точку.
addPoint(FunctionPoint point) - добавляет точку.
Для копирование массивов будем использовать System.arraycopy()

```
// Удаление точки
  public void deletePoint(int index){
        if (points.length <= 2) { // Нельзя удалить, если точек меньше 3
                return;
        }
        FunctionPoint[] newPoints = new FunctionPoint[points.length - 1];
        System.arraycopy(points, 0, newPoints, 0, index);
        System.arraycopy(points, index + 1, newPoints, index, points.length - index - 1);
        points = newPoints;
    }

// Добавление точки
  public void addPoint(FunctionPoint point) {
        FunctionPoint[] newPoints = new FunctionPoint[points.length+ 1];
        int i= 0;
        while (i< points.length && point.getX() > points[i].getX()) {
            i++;
        }
        // Если точка с таким X уже существует - выходим
        if (i < points.length && point.getX() == points[i].getX()){
            return;
        }
        // Копируем старый массив, вставляя новую точку в нужную позицию
        System.arraycopy(points, 0, newPoints, 0, i);
        newPoints[i] = new FunctionPoint(point);
        System.arraycopy(points, i, newPoints, i + 1, points.length-i);

        points = newPoints;
   }




}
```

Задание 7:

Я решил взять функцию квадратного корня, с изначальным количеством точек 11, с диапазоном от 0 до 1. Напишем main и проверим работоспособность программы, а точнее будем проверять правильно ли выводятся значения исходных точек, найдем значение случайных разных точек, изменим точки по y и по x,

```
import functions.*;

public class Main {
    public static void main(String[] args) {
        // Создаем массив значений для функции квадратного корня y=√x
        double[] sqrtValues = new double[11];
        float x;
        for (int i = 0; i < 11; i++) {
            x = (float)i / 10;
            sqrtValues[i] = Math.sqrt(x);
        }

        TabulatedFunction sqrtFunction = new TabulatedFunction(0.0, 1.0, sqrtValues);


        // Выводим информацию о функции
        System.out.println("Левая граница области определения: " + sqrtFunction.getLeftDomainBorder());
        System.out.println("Правая граница области определения: " + sqrtFunction.getRightDomainBorder());
        System.out.println("Количество точек: " + sqrtFunction.getPointsCount());

        // Выводим все точки функции
        System.out.println("\nВсе точки функции:");
        for (int i = 0; i < sqrtFunction.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": x=" + sqrtFunction.getPointX(i) +
                               ", y=" + sqrtFunction.getPointY(i));
        }

        // Различные точки
        System.out.println("\n   ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ   ");
        double[] testPoints = {-0.5, 0.0, 0.05, 0.15, 0.25, 0.35, 0.55, 0.75, 0.95, 1.0, 1.5};

        for (double point : testPoints) {
            double value = sqrtFunction.getFunctionValue(point);
            System.out.println("f(" + point + ") = " + value);
        }
```

добавим и удалим точки, проверим копирование и попробуем выйти за границы.

```
        // Изменяем точки
        System.out.println("\n   ИЗМЕНЯЕМ ТОЧКИ   ");
        System.out.println("До изменения - Точка 5: x=" + sqrtFunction.getPointX(5) +
                           ", y=" + sqrtFunction.getPointY(5));
        sqrtFunction.setPointY(5, 0.8);
        System.out.println("После изменения Y - Точка 5: x=" + sqrtFunction.getPointX(5) +
                           ", y=" + sqrtFunction.getPointY(5));

        // Попробуем изменить X (сработает если не нарушается порядок)
        double oldX = sqrtFunction.getPointX(3);
        sqrtFunction.setPointX(3, 0.32);
        System.out.println("После попытки изменения X точки 3: " + sqrtFunction.getPointX(3));

        // Добавляем точку
        System.out.println("\n ДОБАВЛЯЕМ ТОЧКИ ");
        System.out.println("Количество точек до добавления: " + sqrtFunction.getPointsCount());
        FunctionPoint newPoint = new FunctionPoint(0.45, Math.sqrt(0.45));
        sqrtFunction.addPoint(newPoint);
        System.out.println("Количество точек после добавления: " + sqrtFunction.getPointsCount());

        // Найдем добавленную точку
        for (int i = 0; i < sqrtFunction.getPointsCount(); i++) {
            if (Math.abs(sqrtFunction.getPointX(i) - 0.45) < 0.001) {
                System.out.println("Координаты добавленной точки: x=" + sqrtFunction.getPointX(i) +
                                   ", y=" + sqrtFunction.getPointY(i));
            }
        }

        // Тестируем удаление точки
        System.out.println("\n   УДАЛЯЕМ ТОЧКУ   ");
        System.out.println("Количество точек до удаления: " + sqrtFunction.getPointsCount());
        sqrtFunction.deletePoint(2);
        System.out.println("Количество точек после удаления: " + sqrtFunction.getPointsCount());


        // Проверяем граничные случаи
        System.out.println("\n. ГРАНИЦЫ  ");
        System.out.println("f(-0.1) = " + sqrtFunction.getFunctionValue(-0.1));
        System.out.println("f(1.1) = " + sqrtFunction.getFunctionValue(1.1));

        // Проверяем копирование точки
        System.out.println("\n ПРОВЕРКА КОПИРОВАНИЯ ТОЧКИ   ");
        FunctionPoint original = new FunctionPoint(0.5, 0.707);
        FunctionPoint copy = new FunctionPoint(original);
        System.out.println("Оригинал: x=" + original.getX() + ", y=" + original.getY());
        System.out.println("Копия: x=" + copy.getX() + ", y=" + copy.getY());
    }
}
```

Вывод консоли на другой странице:

```
vladislavvalinevich@MacBook-Air-Vladislav Lab 2 % javac Main.java
vladislavvalinevich@MacBook-Air-Vladislav Lab 2 % java Main
Левая граница области определения: 0.0
Правая граница области определения: 1.0
Количество точек: 11

Все точки функции:
Точка 0: x=0.0, y=0.0
Точка 1: x=0.1, y=0.3162277683729184
Точка 2: x=0.2, y=0.4472135988319589
Точка 3: x=0.30000000000000004, y=0.5477225683874355
Точка 4: x=0.4, y=0.6324555367458368
Точка 5: x=0.5, y=0.7071067811865476
Точка 6: x=0.6000000000000001, y=0.7745966846313364
Точка 7: x=0.7000000000000001, y=0.8366600194099578
Точка 8: x=0.8, y=0.8944271976639178
Точка 9: x=0.9, y=0.9486832854847512
Точка 10: x=1.0, y=1.0

   ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ
f(-0.5) = NaN
f(0.0) = 0.0
f(0.05) = 0.1581138841864592
f(0.15) = 0.3817206836024386
f(0.25) = 0.49746808360969713
f(0.35) = 0.5900890525666361
f(0.55) = 0.7408517329089419
f(0.75) = 0.8655436085369378
f(0.95) = 0.9743416427423756
f(1.0) = 1.0
f(1.5) = NaN

   ИЗМЕНЯЕМ ТОЧКИ
До изменения - Точка 5: x=0.5, y=0.7071067811865476
После изменения Y - Точка 5: x=0.5, y=0.8
После попытки изменения X точки 3: 0.32

 ДОБАВЛЯЕМ ТОЧКИ
Количество точек до добавления: 11
Количество точек после добавления: 12
Координаты добавленной точки: x=0.45, y=0.6708203932499369

   УДАЛЯЕМ ТОЧКУ
Количество точек до удаления: 12
Количество точек после удаления: 11

. ГРАНИЦЫ
f(-0.1) = NaN
f(1.1) = NaN

 ПРОВЕРКА КОПИРОВАНИЯ ТОЧКИ
Оригинал: x=0.5, y=0.707
Копия: x=0.5, y=0.707
vladislavvalinevich@MacBook-Air-Vladislav Lab 2 %
```

Вывод: Я разработал набор классов для работы с функциями одной переменной, заданными в табличной форме.