

ЛАБОРАТОРНАЯ РАБОТА №2

Курс: Объектно-ориентированное
программирование.

Студент: Кузнецов Эрик Витальевич

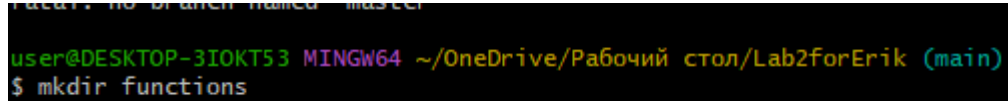
Группа:6204-010302D

Преподаватель: Борисов Дмитрий Сергеевич

Задание 1

Ход выполнения работы:

В ходе первого задания было необходимо создать пакет functions, в котором далее будут создаваться классы программы.



```
fatal: no branch named 'main'  
user@DESKTOP-3I0KT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik (main)  
$ mkdir functions
```

Рисунок 1-изображение с конечным результатом для задания 1.

Задание 2.

Ход выполнения работы:

- 1) папке functions создаем файл FunctionPoint.java
- 2) В классе объявляем два частных поля: x и y типа double для обеспечения инкапсуляции
- 3) Реализуем конструкторы в соответствии с заданием
- 4) Добавляем методы доступа (геттеры) для полей x и y.

```
user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik (main)
$ cd functions

user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik/functions (main)
$ nano FunctionPoint.java
```

```
GNU nano 8.5
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.x=point.x;
        this.y=point.y;
    }
    public FunctionPoint(){
        this.x=this.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;
    }
}
```

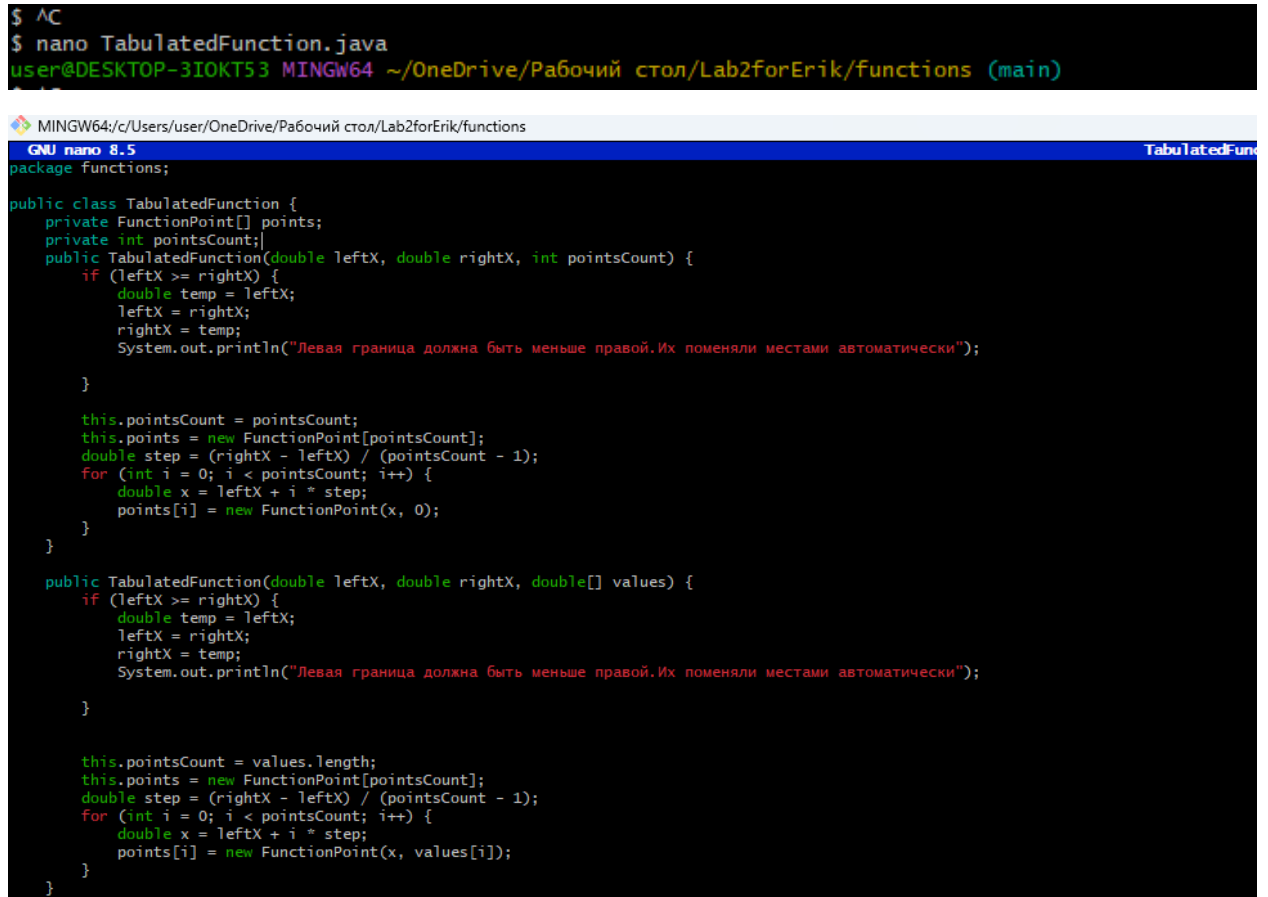
Рисунок 2,3-Снимок Экрана с конечным результатом для задания 2

Задание 3

Ход выполнения работы:

1) В папке functions создаем файл TabulatedFunction.

2) Выполняем все согласно заданию.



```
$ AC
$ nano TabulatedFunction.java
user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik/functions (main)

MINGW64/c:/Users/user/OneDrive/Рабочий стол/Lab2forErik/functions
GNU nano 8.5 TabulatedFun
package functions;

public class TabulatedFunction {
    private FunctionPoint[] points;
    private int pointsCount;
    public TabulatedFunction(double leftX, double rightX, int pointsCount) {
        if (leftX >= rightX) {
            double temp = leftX;
            leftX = rightX;
            rightX = temp;
            System.out.println("Левая граница должна быть меньше правой. Их поменяли местами автоматически");
        }

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

    public TabulatedFunction(double leftX, double rightX, double[] values) {
        if (leftX >= rightX) {
            double temp = leftX;
            leftX = rightX;
            rightX = temp;
            System.out.println("Левая граница должна быть меньше правой. Их поменяли местами автоматически");
        }

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

Рисунок 3,4-Снимок Экрана с конечным результатом для задания 3.

Задание 4

Ход выполнения работы:

1) В классе TabulatedFunction описываем методы, необходимые для работы с функцией. (double getLeftDomainBorder(), double getRightDomainBorder(), double getFunctionValue(double x))

```
public double getLeftDomainBorder() {  
    return points[0].getX();  
}  
  
public double getRightDomainBorder() {  
    return points[pointsCount - 1].getX();  
}  
  
public double getFunctionValue(double x) {  
    if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {  
        return Double.NaN; //Если x не в диапазоне ,то возвращаем Nan  
    }  
    for (int i = 0; i < pointsCount - 1; i++) { //Берем каждый минимальный отрезок между объектами, берем их X  
        double x1 = points[i].getX();  
        double x2 = points[i + 1].getX();  
  
        if (x >= x1 && x <= x2) { //Если исходный x между ними ,то находим для него y по формуле  
            double y1 = points[i].getY();  
            double y2 = points[i + 1].getY();  
  
            return y1 + (y2 - y1) * (x - x1) / (x2 - x1);  
        }  
    }  
    return Double.NaN;  
}
```

Рисунок 5-Снимок Экрана с конечным результатом для задания 4.

Задание 5

Ход выполнения работы:

- 1) В классе TabulatedFunction описываем методы, необходимые для работы с точками табулированной функции. Методы: int getPointsCount, FunctionPoint getPoint, void setPoint, double getPointX, void setPointX, double getPointY, void setPointY.

```
public int getPointsCount(){
    return pointsCount;
}
public FunctionPoint getPoint(int index) {
    return new FunctionPoint(points[index]);
}
public void setPoint(int index, FunctionPoint point) {
    if (index > 0 && point.getX() <= points[index - 1].getX()) {
        return;
    }
    if (index < pointsCount - 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 < pointsCount - 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);
}
```

Рисунок 6-Снимок Экрана с конечным результатом для задания 5.

Задание 6

Ход выполнения работы:

- 1) В классе TabulatedFunction описываем методы, изменяющие количество точек табулированной функции. Методы: void deletePoint и void addPoint.

```
}  
public void deletePoint(int index) {  
    for (int i = index; i < pointsCount - 1; i++) {  
        points[i] = points[i + 1];  
    }  
    --pointsCount;  
}  
public void addPoint(FunctionPoint point){  
    int count=0;  
    while(count < pointsCount && point.getXO()>points[count].getXO()){//проверка текущего X объекта с X объекта point  
        count++;  
    }  
    if (count < pointsCount && point.getXO()==points[count].getXO()){//Проверка на равенство X ,если так ,то вставлять элемент не будем  
        return ;  
    }  
    // Проверяем необходимость увеличения массива  
    if (pointsCount >= points.length) {  
        FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];  
        for (int i = 0; i < pointsCount; i++) {  
            newPoints[i] = points[i];  
        }  
        points = newPoints;  
    }  
    // Сдвигаем элементы ,Выходит так что на месте point[count] и points[count+1] стоят два одинаковых элемента  
    for (int i = pointsCount; i > count; i--) {  
        points[i] = points[i - 1];  
    }  
    // Вставляем новую точку  
    points[count] = new FunctionPoint(point);  
    // увеличиваем счетчик  
    pointsCount++;  
}  
}
```

Рисунок 7-Снимок экрана с конечным результатом для задания 6.

Задание 7

Ход выполнения работы:

- 1) В пакете по умолчанию (вне пакета functions) создаем класс Main, содержащий точку входа программы.
- 2) В методе main() создаем экземпляр класса TabulatedFunction и задаем для него табулированные значения функции: $y=2x+1$
- 3) Выводим в консоль значения функции на ряде точек. А также выводим результат после того, как изменим, добавим или удалим некоторые точки.

```
user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik/functions (main)
$ cd ..

user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik (main)
$ nano Main.java

user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik (main)
$ javac Main.java

user@DESKTOP-3IOKT53 MINGW64 ~/OneDrive/Рабочий стол/Lab2forErik (main)
$ java Main
1. Создание функции  $y = 2x+1$  на интервале  $[0, 10]$  с 6 точками

Вычисление значений функции в разных точках:
f(-1.0) = не определена (вне области определения)
f(0.0) = 1.0
Такое значение X уже было
f(2.0) = 5.0
f(4.0) = 9.0
f(6.0) = 13.0
f(8.0) = 17.0
f(10.0) = 21.0

Границы области определения:
Левая граница: 0.0
Правая граница: 10.0

Исходные точки функции:
Точка 0: (0.0 1.0)
Точка 1: (2.0 5.0)
Точка 2: (4.0 9.0)
Точка 3: (6.0 13.0)
Точка 4: (8.0 17.0)
Точка 5: (10.0 21.0)

2. Изменение точек функции:
Изменение первой точки (индекс 0) на (0,100):
Точка 0: (0.0 100.0)
Точка 1: (2.0 5.0)
Точка 2: (4.0 9.0)
Точка 3: (6.0 13.0)
Точка 4: (8.0 17.0)
Точка 5: (10.0 21.0)

3. Добавление новой точки (6.6, 9.9):
Количество точек до добавления: 6
Количество точек после добавления: 7

Новые точки функции после добавления:
Точка 0: (0.0 100.0)
Точка 1: (2.0 5.0)
Точка 2: (4.0 9.0)
Точка 3: (6.0 13.0)
Точка 4: (6.6 9.9)
Точка 5: (8.0 17.0)
Точка 6: (10.0 21.0)

4. Удаляем точку с индексом 3:
Точка 0: (0.0 100.0)
Точка 1: (2.0 5.0)
Точка 2: (4.0 9.0)
Точка 3: (6.6 9.9)
Точка 4: (8.0 17.0)
Точка 5: (10.0 21.0)
```

Рисунок 8,9-Снимок экрана с конечным результатом для задания 7