

Лабораторная работа № 2

Выполнила: Сингатуллина Алина Марсовна

Группа 6204-010302D

Оглавление

Задание 1	3
Задание 2	3
Задание 3.....	4
Задание 4.....	5
Задание 5.....	6
Задание 6.....	7
Задание 7.....	7
Вывод.....	8

Задание 1

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

Задание 2

В пакете `functions` создала класс `FunctionPoint`, объект которого описывает одну точку табулированной функции. Состояние объектов содержит два аспекта: координату точки по оси абсцисс и координату точки по оси ординат.

В классе я описала 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).

FunctionPoint.java

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

    public FunctionPoint(double x, double y) { // конструктор с параметрами
        this.x = x;
        this.y = y;
    }

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

    public FunctionPoint() { // конструктор по умолчанию
        x = 0;
        y = 0;
    }

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

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

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

```

    }

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

```

Задание 3

Я создала в пакете functions класс TabulatedFunction, его объект описывает табулированную функцию. Для хранения данных о точках должен использовать массив типа FunctionPoint.

Далее в классе описала следующие конструкторы:

- 1) 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;
- 2) TabulatedFunction(double leftX, double rightX, double[] values) – аналогичен предыдущему конструктору, но вместо количества точек получает значения функции в виде массива.

В данных случаях точки создаются через равные интервалы по x.

TabulatedFunction.java

```

package functions;

public class TabulatedFunction
{
    private FunctionPoint[] mas;
    private int count;

    public TabulatedFunction(double leftX, double rightX, int pointsCount){//конструктор с кол
ом точек
        count = pointsCount;
        mas = new FunctionPoint[pointsCount];
        double step = (rightX - leftX)/(pointsCount - 1);
        double x = leftX;

        for(int i = 0; i < pointsCount; ++i)
        {
            mas[i] = new FunctionPoint(x, 0);
            x+=step;
        }
    }

    public TabulatedFunction(double leftX, double rightX, double[] values)//конструктор с
массивом значений
    {
        count = values.length;
        mas = new FunctionPoint[values.length];
    }
}

```

```

double step = (rightX - leftX)/(values.length - 1);
double x = leftX;

for(int i = 0; i < values.length; ++i)
{
    mas[i] = new FunctionPoint(x, values[i]);
    x+=step;
}
}

```

Задание 4

Реализовала методы для работы с функцией:

- 1) getLeftDomainBorder() - возвращает левую границу области определения
 - 2) getRightDomainBorder() - возвращает правую границу области определения
 - 3) getFunctionValue(double x) - вычисляет значение функции в точке x с использованием линейной интерполяции
- Метод getFunctionValue() возвращает Double.NaN для точек вне области определения.

```

public double getLeftDomainBorder()//левая граница области определения
{
    return this.mas[0].getX();
}

public double getRightDomainBorder()//правая граница области определения
{
    return this.mas[count-1].getX();
}

public double getFunctionValue(double x) //значение функции в точке
{
    if (x >= getLeftDomainBorder() && x <= getRightDomainBorder())
    {
        int left = 0;
        int right = this.count - 1;
        int middle = (left+right)/2;
        while(left<=right)
        {
            middle = (left+right)/2;
            if(this.mas[middle].getX()==x)
                return mas[middle].getY();
            else
            if(this.mas[middle].getX() < x){
                left = middle+1;
            }
            else
                right = middle-1;
        }
        if(mas[middle].getX()<x)
            ++middle;
    }
}

```

```

        double k = (this.mas[middle].getY() - this.mas[middle - 1].getY()) / (this.mas[middle].getX()
- this.mas[middle - 1].getX());
        double b = this.mas[middle].getY() - k * this.mas[middle].getX();

        return k * x + b;
    }
    else
        return Double.NaN;
}

```

Задание 5

Добавила методы для работы с точками функции:

- 1) `getPointsCount()` - количество точек
- 2) `getPoint(int index)` - возвращает копию точки (инкапсуляция)
- 3) `setPoint(int index, FunctionPoint point)` - заменяет точку с проверкой порядка
- 4) Геттеры и сеттеры для координат x и y
- 5) Метод `setPoint()` проверяет, что новая координата x не нарушает порядок точек.

```

public int getPointsCount() //количество точек
{
    return this.count;
}

public FunctionPoint getPoint(int index) //получение копии точки
{
    FunctionPoint j = new FunctionPoint(mas[index]);
    return j;
}

public void setPoint(int index, FunctionPoint point) //установка точки
{
    if (point.getX() >= getLeftDomainBorder() && point.getX() <=
getRightDomainBorder())
    {
        mas[index].setX(point.getX());
        mas[index].setY(point.getY());
    }
}

//геттеры и сеттеры для координат X и Y
public double getPointX(int index)
{
    return this.mas[index].getX();
}

public void setPointX(int index, double x)
{
    if (x >= getLeftDomainBorder() && x <= getRightDomainBorder())
        this.mas[index].setX(x);
}

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

public void setPointY(int index, double y)
{
    if (y >= getLeftDomainBorder() && y <= getRightDomainBorder())

```

```

        this.mas[index].setY(y);
    }
}

```

Задание 6

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

- 1) Метод void deletePoint(int index) удаляет заданную точку табулированной функции.
- 2) Метод void addPoint(FunctionPoint point) добавляет новую точку табулированной функции. При написании метода обеспечьте корректную инкапсуляцию.

Реализованы эффективные методы модификации табличной функции.

Задание 7

В пакете по умолчанию я создала класс Main, содержащий точку входа программы. В методе main() сделала экземпляр класса TabulatedFunction и задала для него табулированные значения функции $f(x) = x^2$.

```

import functions.*;

public class Main {

    public static void main(String[] args) {
        double[] x2 = {9, 4, 1, 0, 1, 4, 9};
        TabulatedFunction f = new TabulatedFunction(-3, 3, x2);
        for(int i = 0; i < f.getPointsCount(); i++)
        {
            System.out.println("X= " + f.getPointX(i) + "   Y= " + f.getPointY(i));
        }

        System.out.println(f.getLeftDomainBorder() + " " +
f.getRightDomainBorder());

        System.out.println("x = -2   " + f.getFunctionValue(-2));
        System.out.println("x = -4   " + f.getFunctionValue(-4));
        System.out.println(f.getPointsCount());
        System.out.println(f.getPointX(5));
        f.deletePoint(5);
        for(int i = 0; i < f.getPointsCount(); i++)
        {
            System.out.println("X= " + f.getPointX(i) + "   Y= " + f.getPointY(i));
        }
        System.out.println("////////////////////");
        FunctionPoint p = new FunctionPoint(-2.5, 6);
        f.addPoint(p);
        for(int i = 0; i < f.getPointsCount(); i++)
        {
            System.out.println("X= " + f.getPointX(i) + "   Y= " + f.getPointY(i));
        }
        System.out.println("////////////////////");
        FunctionPoint o = new FunctionPoint(-2.1, 6.2);
        f.setPoint(1, o);
        for(int i = 0; i < f.getPointsCount(); i++)
        {
            System.out.println("X= " + f.getPointX(i) + "   Y= " + f.getPointY(i));
        }
    }
}

```

Результаты запуска Main.java

```
x= -3.0  Y= 9.0
x= -2.0  Y= 4.0
x= -1.0  Y= 1.0
x= 0.0   Y= 0.0
x= 1.0   Y= 1.0
x= 2.0   Y= 4.0
x= 3.0   Y= 9.0
-3.0 3.0
x = -2   4.0
x = -4   NaN
7
2.0
x= -3.0  Y= 9.0
x= -2.0  Y= 4.0
x= -1.0  Y= 1.0
x= 0.0   Y= 0.0
x= 1.0   Y= 1.0
x= 3.0   Y= 9.0
//////////
x= -3.0  Y= 9.0
x= -2.5  Y= 6.0
x= -2.0  Y= 4.0
x= -1.0  Y= 1.0
x= 0.0   Y= 0.0
x= 1.0   Y= 1.0
x= 3.0   Y= 9.0
//////////
x= -3.0  Y= 9.0
x= -2.1  Y= 6.2
x= -2.0  Y= 4.0
x= -1.0  Y= 1.0
x= 0.0   Y= 0.0
x= 1.0   Y= 1.0
x= 3.0   Y= 9.0
```

Создается функция на отрезке $[-3, 3]$ с 7 точками. Программа тестирует все основные операции с табулированной функцией: создание функции, чтение точек, вычисление значений, удаление, добавление, изменение точек, работа с границами области определения

Вывод

В ходе лабораторной работы успешно разработана система классов для работы с табличными функциями. Реализованы все требуемые методы с соблюдением принципов ООП и инкапсуляции. Программа протестирована и работает корректно.