Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «ООП»

Тема «Работа с функциями одной переменной, заданными в табличной форме»

Выполнил: Куликов Степан
Дмитриевич

Группа: 6201-120303D

Самара, 2025

Задание на лабораторную работу

Задание 1

Создать пакет functions, в котором далее будут создаваться классы программы.

Задание 2

В пакете functions создать класс FunctionPoint, объект которого должен описывать одну точку табулированной функции.

Ход выполнения задания:

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

Создал два закрытых поля x и y – инкапсуляция

```
public FunctionPoint(double x, double y){…}
```

Конструктор инициализирующий объект с заданными координатами

```
public FunctionPoint(FunctionPoint point){…}
```

Конструктор, создаёт новую точку (копию) с теми же координатами, что и у переданной

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

Создание точки по умолчанию (0;0)

```
public double getX(), setX(), getY(), setY()
```

Обеспечение доступа к значениям

Задание 3

В пакете functions создать класс TabulatedFunction, объект которого должен описывать табулированную функцию.

Ход выполнения задания:

private FunctionPoint[] point;
private int pointsCount;

Массив point хранит точки функции, pointsCount количество точек.

public TabulatedFunction(double leftX, double rightX, int pointsCount)

Конструктор создает табулированную функцию на интервале от leftX до rightX, деля его на равные отрезки, все значения y равны 0

double prom = (rightX-leftX)/(pointsCount-1);

Шаг между x

public TabulatedFunction(double leftX, double rightX, double[] values)

Конструктор создаёт функцию на интервале и задаёт y из массива values, получается таблица точек (x;y) с равным интервалом между x

Задание 4

В классе TabulatedFunction описать методы, необходимые для работы с функцией.

Ход выполнения работы:

public double getLeftDomainBorder()
public double getRightDomainBorder()

Методы возвращающие границы интервала по x

public double getFunctionValue(double x)

Метод вычисления значения функции: проверяет находится ли x в области определения, если x совпадает с одной из табличных точек, то возвращает точное y, если x между двумя точками высчитывает y и вставляет между этими точками. Если x вне области определения возвращает Double.NaN.

Задание 5

В классе TabulatedFunction описать методы, необходимые для работы с точками табулированной функции

Ход выполнения работы:

getPointsCount()

Возвращает количество точек

getPoint(index)

Возвращает копию точки

setPoint(index, FunctionPoint p)

Заменяет точку, если сохраняется порядок X

getPointX/Y() и setPointX/Y()

Доступ к координатам по отдельности

Задание 6

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

Ход выполнения работы:

public void deletePoint(int index)

Удаляет точку с указанным индексом, переносит элементы через System.arraycopy()

public void addPoint(FunctionPoint p)

Создаёт новый массив на одну позицию больше, вставляет точку в нужное место.

Задание 7

Проверить работу написанных классов.

Ход выполнения работы:

```
double left = 0;
double right = 4;
double[] values = {0, 1, 4, 9, 16};
```

Задали границы и исходную функцию

```
TabulatedFunction f = new TabulatedFunction(left, right, values);
```

Создал экземпляр класса и задал для него значения

```
double[] testX = {-1, 0, 1.5, 2, 2.5, 3.7, 5};
for (double x : testX) {
   System.out.println("f(" + x + ") = " + f.getFunctionValue(x));
}
```

Вычислил значение функции в разных x

```
f.addPoint(new FunctionPoint(2.5, 6.25));
for (int i = 0; i < f.getPointsCount(); i++) {
   System.out.println("(" + f.getPoint(i).getX() + ", " + f.getPoint(i).getY() + ")");
}


f.deletePoint(1);
for (int i = 0; i < f.getPointsCount(); i++) {
   System.out.println("(" + f.getPoint(i).getX() + ", " + f.getPoint(i).getY() + ")");
}
```

Добавил и удалил точку

```
for (double x : testX) {
   System.out.println("f(" + x + ") = " + f.getFunctionValue(x));
}
```

Повторно вычислил значение функции в разных x

Команды в консоли:

C:\Users\fael7\Desktop\лр2\Lab-2-2025>javac Main.java

C:\Users\fael7\Desktop\лр2\Lab-2-2025>java Main

Исходная табулированная функция:

(0.0, 0.0)

(1.0, 1.0)

(2.0, 4.0)

(3.0, 9.0)

(4.0, 16.0)

Проверка значений функции:

f(-1.0) = NaN

f(0.0) = 0.0

f(1.5) = 2.5

f(2.0) = 4.0

f(2.5) = 6.5

f(3.7) = 13.900000000000002

f(5.0) = NaN

Добавляем новую точку (2.5, 6.25):

(0.0, 0.0)

(1.0, 1.0)

(2.0, 4.0)

(2.5, 6.25)

(3.0, 9.0)

(4.0, 16.0)

Удаляем точку с индексом 1:

(0.0, 0.0)

(2.0, 4.0)

(2.5, 6.25)

(3.0, 9.0)

(4.0, 16.0)

Повторная проверка после изменений:

f(-1.0) = NaN

f(0.0) = 0.0

f(1.5) = 3.0

f(2.0) = 4.0

f(2.5) = 6.25

f(3.7) = 13.900000000000002

f(5.0) = NaN

C:\Users\fael7\Desktop\лр2\Lab-2-2025>git add .

C:\Users\fael7\Desktop\лр2\Lab-2-2025>git commit -m "Lab 2"

[main 085facf] Lab 2

6 files changed, 183 insertions(+)

create mode 100644 Main.class

```
 create mode 100644 Main.java
 create mode 100644 functions/FunctionPoint.class
 create mode 100644 functions/FunctionPoint.java
 create mode 100644 functions/TabulatedFunction.class
 create mode 100644 functions/TabulatedFunction.java

C:\Users\fael7\Desktop\лр2\Lab-2-2025>git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 5.33 KiB | 606.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/HannahMontana2006/Lab-2-2025.git
   d6809c3..085facf  main -> main
```