# Отчёт
# Лабораторная работа №2

Выполнил: Салихов Камран Рустам угли
Группа: 6204-010302D

В ходе выполнения задания 1 был создан пакет **functions**.

В ходе выполнения задания 2 в пакете **functions** был создан класс **FunctionPoint,** объект которого описывает одну точку табулированной функции.

В классе были описаны следующие конструкторы:

```
public FunctionPoint() {x = 0; y = 0;}  no usages

public FunctionPoint(double x, double y) {this.x = x; this.y = y;}  4 usages

public FunctionPoint(FunctionPoint point) {x = point.x; y = point.y;}  3 usages
```

- создаёт точку с координатами (0; 0)

- создаёт объект точки с заданными координатами

- создаёт объект точки с теми же координатами, что у указанной точки

В ходе выполнения задания 3 в пакете **functions** был создан класс **TabulatedFunction,** объект которого описывает табулированную функцию. Массив **FunctionPoint[] points** используется для хранения точек. Переменная **size** - для отслеживания количества точек.

В классе описаны следующие конструкторы:

**public 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

**public TabulatedFunction(double leftX, double rightX, double[] values)** - аналогичен предыдущему конструктору, но вместо количества точек получает значения функции в виде массива

В ходе выполнения задания 4 в классе **TabulatedFunction** были описаны методы, необходимые для работы с функцией:
**public double getLeftDomainBorder()** - возвращает левую границу области определения

**public double getRightDomainBorder()** - возвращает правую границу области определения

**public double getFunctionValue(double x)** - вычисляет значение функции в точке x

При расчёте значения функции используется линейная интерполяция.

```
public double getFunctionValue(double x) {  no usages
    if ((size == 0) || (x < getLeftDomainBorder()) || x > getRightDomainBorder()) return Double.NaN;

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

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

В ходе выполнения задания 5 в классе **TabulatedFunction** были описаны методы, необходимые для работы с точками табулированной функции:

**public int getPointsCount()** - возвращает количество точек

**public FunctionPoint getPoint(int index)** - возвращает копию точки

**public void setPoint(int index, FunctionPoint point)** - заменяет точку с проверкой упорядоченности

**public double getPointX(int index), public void setPointX(int index, double x)** - работа с координатой X

**public double getPointY(int index), public void setPointY(int index, double y)** - работа с координатой Y

Инкапсуляция:

```
public FunctionPoint getPoint(int index) {  no usages
    if (index >= 0 && index < size) return new FunctionPoint(points[index]);
    else return null;
}
```

В ходе выполнения задания 6 в классе **TabulatedFunction** были описаны методы, изменяющие количество точек табулированной функции:

```
public void deletePoint(int index) {  1 usage
    if (index >= 0 && index < size) {
        --size;
        System.arraycopy(points,  srcPos: index + 1, points, index,  length: size - index);
    }
}
```

- удаляется заданная точка табулированной функции, используется **System.arraycopy()**, сохраняется упорядоченность массива

**public void addPoint(FunctionPoint point)** - добавляется новая точка табулированной функции, создается копия точки для инкапсуляции, находится позиция для вставки, сохраняется упорядоченность, при необходимости расширяется массив, точка вставляется в найденную позицию

В ходе выполнения задания 7 был создан класс **Main** вне пакета **functions**. Результат работы программы при введенных параметрах:

начальные (-1.0, 4.0, 6)

замена (2.0, 5.0), индекс замены (2)

добавление (-3.5, 0.75)

индекс удаления (0)

```
начальные точки:
x = -1.0 y = 0.0
x = 0.0 y = 0.0
x = 1.0 y = 0.0
x = 2.0 y = 0.0
x = 3.0 y = 0.0
x = 4.0 y = 0.0
после замены:
x = -1.0 y = 0.0
x = 0.0 y = 0.0
x = 2.0 y = 5.0
x = 2.0 y = 0.0
x = 3.0 y = 0.0
x = 4.0 y = 0.0
после добавления:
x = -3.5 y = 0.75
x = -1.0 y = 0.0
x = 0.0 y = 0.0
x = 2.0 y = 5.0
x = 2.0 y = 0.0
x = 3.0 y = 0.0
x = 4.0 y = 0.0
после удаления:
x = -1.0 y = 0.0
x = 0.0 y = 0.0
x = 2.0 y = 5.0
x = 2.0 y = 0.0
x = 3.0 y = 0.0
x = 4.0 y = 0.0
```

начальные (-2.0, 4.5, 5)

замена (2.0, 5.0), индекс замены (3)

добавление (-3.5, 0.75)

индекс удаления (3)

```
начальные точки:
x = -2.0 y = 0.0
x = -0.375 y = 0.0
x = 1.25 y = 0.0
x = 2.875 y = 0.0
x = 4.5 y = 0.0
после замены:
x = -2.0 y = 0.0
x = -0.375 y = 0.0
x = 1.25 y = 0.0
x = 2.0 y = 5.0
x = 4.5 y = 0.0
после добавления:
x = -3.5 y = 0.75
x = -2.0 y = 0.0
x = -0.375 y = 0.0
x = 1.25 y = 0.0
x = 2.0 y = 5.0
x = 4.5 y = 0.0
после удаления:
x = -3.5 y = 0.75
x = -2.0 y = 0.0
x = -0.375 y = 0.0
x = 2.0 y = 5.0
x = 4.5 y = 0.0
```