# Отчёт по лабораторной работе №2

**Разработка классов для работы с табулированными функциями**

**ФИО:** Пожидаев Богдан Витальевич
**Группа:** 6204-010302D

**Задание 1-2**

**Создание пакета functions и класса FunctionPoint**

Реализован класс FunctionPoint для описания точки табулированной функции. Класс включает:

- Поля x и y с модификатором доступа private
- Конструкторы:

  FunctionPoint(double x, double y)

  FunctionPoint(FunctionPoint point) (копирующий)

  FunctionPoint() (по умолчанию, создаёт точку (0; 0))
- Геттеры и сеттеры для обеспечения инкапсуляции

```
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 = 0;
        this.y = 0;
    }

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

    public double getX() {
        return x;
    }

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

    public double getY() {
        return y;
    }
}
```

**Задание 3-6**

**Создание класса TabulatedFunction**

Реализован класс для работы с табулированной функцией, хранящей точки в массиве FunctionPoint[], упорядоченном по возрастанию x.

**Конструкторы:**

- TabulatedFunction(double leftX, double rightX, int pointsCount) – создаёт точки с равным шагом, значения y = 0
- TabulatedFunction(double leftX, double rightX, double[] values) – создаёт точки с заданными значениями y

**Методы работы с функцией:**

- getLeftDomainBorder(), getRightDomainBorder() – возвращают границы области определения
- getFunctionValue(double x) – вычисляет значение функции с использованием линейной интерполяции

**Методы работы с точками:**

- getPointsCount() – возвращает количество точек
- getPoint(int index) – возвращает копию точки
- setPoint(int index, FunctionPoint point) – заменяет точку с проверкой порядка
- Методы get/setPointX() и get/setPointY() с соответствующими проверками

**Методы изменения количества точек:**

- deletePoint(int index) – удаляет точку, сохраняя порядок
- addPoint(FunctionPoint point) – добавляет точку в правильную позицию с использованием System.arraycopy()

```
package functions;

public class TabulatedFunction {
    private FunctionPoint[] points;
    private int pointsCount;

    // конст создания объекта табул. ф-и через кол-во точек
    public TabulatedFunction(double leftX, double rightX, int
pointsCount) {
        if (pointsCount < 2) pointsCount = 2;
        this.pointsCount = pointsCount;
        this.points = new FunctionPoint[pointsCount + 5]; // Запас
памяти

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

        for (int i = 0; i < pointsCount; i++) {
```

```
            double x = leftX + step * i;
            points[i] = new FunctionPoint(x, 0);
        }
    }

    // конст. созд. объекта через массив объектов
    public TabulatedFunction(double leftX, double rightX, double[]
values) {
        this.pointsCount = values.length;
        this.points = new FunctionPoint[values.length + 5];

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

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

    public double getLeftDomainBorder() {
        return points[0].getX();
    }

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

    // значение ф-и в точке x
    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }

        //интервал содерж. x
        for (int i = 0; i < pointsCount - 1; i++) {
            double x1 = points[i].getX();
            double x2 = points[i + 1].getX();

            if (x >= x1 && x <= x2) {
                double y1 = points[i].getY();
                double y2 = points[i + 1].getY();

                // линейная интерполяция
                return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
            }
        }

        if (x == points[pointsCount - 1].getX()) {
            return points[pointsCount - 1].getY();
        }

        return Double.NaN;
    }

    // кол-во точек
    public int getPointsCount() {
        return pointsCount;
    }

    // копия точки
    public FunctionPoint getPoint(int index) {
        if (index < 0 || index >= pointsCount) return null;
        return new FunctionPoint(points[index]);
    }
```

```
    // замена точки
    public void setPoint(int index, FunctionPoint point) {
        if (index < 0 || index >= pointsCount) return;

        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);
    }

    // значение абсциссы(Ox) точки с заданным номером
    public double getPointX(int index) {
        if (index < 0 || index >= pointsCount) return Double.NaN;
        return points[index].getX();
    }

    // изменение значение абсциссы(Ox) точки с заданным номером
    public void setPointX(int index, double x) {
        if (index < 0 || index >= pointsCount) return;

        if (index > 0 && x <= points[index-1].getX()) { return; }
        if (index < (pointsCount-1) && x >= points[index+1].getX()) {
return; }

        points[index].setX(x);
    }

    // значение ординаты(Oy) точки с некоторым номером
    public double getPointY(int index) {
        if (index < 0 || index >= pointsCount) return Double.NaN;
        return points[index].getY();
    }

    // изменение значение ординаты(Oy) точки с некоторым номером
    public void setPointY(int index, double y) {
        if (index < 0 || index >= pointsCount) return;
        points[index].setY(y);
    }

    // удаление заданной точки табулированной функции
    public void deletePoint(int index) {
        if (pointsCount <= 2 || index < 0 || index >= pointsCount)
return;

        System.arraycopy(points, index+1, points, index, pointsCount-
index-1);
        pointsCount--;
    }

    // доб. новую точку табулированной функции
    public void addPoint(FunctionPoint point) {
        // проверка на необходимость расширения
        if (pointsCount >= points.length) {
            FunctionPoint[] newPoints = new FunctionPoint[points.length
* 2];
            System.arraycopy(points, 0, newPoints, 0, pointsCount);
            points = newPoints;
        }

        int pos = 0;
        while (pos < pointsCount && points[pos].getX() < point.getX())
```

```
{
            pos++;
        }

        // сдвиг вправо
        System.arraycopy(points, pos, points, pos + 1, pointsCount -
pos);

        // добавл новой точки
        points[pos] = new FunctionPoint(point);
        pointsCount++;
    }
}
```

**Задание 7**

**Тестирование работы классов**

Создан класс Main для проверки функциональности:

- Создана функция f(x) = 10x + 10 на интервале [0, 5]
- Протестированы вычисления значений внутри и вне области определения
- Проверены операции модификации, добавления и удаления точек
- Протестирована линейная интерполяция

## Результаты вывода:

=== ТЕСТИРОВАНИЕ КЛАССОВ ТАБУЛИРОВАННОЙ ФУНКЦИИ ===

1. СОЗДАНИЕ ФУНКЦИИ f(x) = 10x + 10 НА ИНТЕРВАЛЕ [0, 5]:

Область определения: [0,0, 5,0]

Количество точек: 6

2. ТОЧКИ ТАБУЛИРОВАННОЙ ФУНКЦИИ:

Точка 0: (0,0; 10,0)

Точка 1: (1,0; 20,0)

Точка 2: (2,0; 30,0)

Точка 3: (3,0; 40,0)

Точка 4: (4,0; 50,0)

Точка 5: (5,0; 60,0)

3. ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ:

f(-2,0) = не определено (вне области определения)

f(0,0) = 10,0 (ожидалось: 10,0)

f(0,5) = 15,0 (ожидалось: 15,0)

f(1,0) = 20,0 (ожидалось: 20,0)

f(1,5) = 25,0 (ожидалось: 25,0)

f(2,0) = 30,0 (ожидалось: 30,0)

f(2,5) = 35,0 (ожидалось: 35,0)

f(3,0) = 40,0 (ожидалось: 40,0)

f(3,5) = 45,0 (ожидалось: 45,0)

f(4,0) = 50,0 (ожидалось: 50,0)

f(4,5) = 55,0 (ожидалось: 55,0)

f(5,0) = 60,0 (ожидалось: 60,0)

f(6,0) = не определено (вне области определения)

4. ТЕСТИРОВАНИЕ МОДИФИКАЦИИ ТОЧЕК:

Изменение ординаты точки с индексом 2 на правильное значение:

До: (2,0; 30,0) -> После: (2,0; 30,0)

Корректное изменение абсциссы с обновлением Y:

До: (1,0; 20,0) -> После: (1,2; 22,0)

5. ТЕСТИРОВАНИЕ ДОБАВЛЕНИЯ ТОЧКИ:

Количество точек до добавления: 6

Количество точек после добавления: 7

Значение в добавленной точке: f(2,2) = 32,0

Точки после добавления:

(0,0; 10,0) (1,2; 22,0) (2,0; 30,0) (2,2; 32,0) (3,0; 40,0) (4,0; 50,0) (5,0; 60,0)

6. ТЕСТИРОВАНИЕ УДАЛЕНИЯ ТОЧКИ:

Количество точек до удаления: 7

Количество точек после удаления: 6

Обновление всех значений Y в соответствии с f(x) = 10x + 10:

Точка 0: (0,0; 10,0)

Точка 1: (1,2; 22,0)

Точка 2: (2,2; 32,0)

Точка 3: (3,0; 40,0)

Точка 4: (4,0; 50,0)

Точка 5: (5,0; 60,0)

7. ПРОВЕРКА ИНТЕРПОЛЯЦИИ ПОСЛЕ ИСПРАВЛЕНИЙ:

f(0,3) = 13,0 (ожидалось: 13,0) СОВПАДАЕТ

f(1,1) = 21,0 (ожидалось: 21,0) СОВПАДАЕТ

f(1,8) = 28,0 (ожидалось: 28,0) СОВПАДАЕТ

f(2,7) = 37,0 (ожидалось: 37,0) СОВПАДАЕТ

f(3,9) = 49,0 (ожидалось: 49,0) СОВПАДАЕТ

f(4,6) = 56,0 (ожидалось: 56,0) СОВПАДАЕТ

8. ГРАНИЧНЫЕ ЗНАЧЕНИЯ:

Левая граница: f(0,0) = 10,0

Правая граница: f(5,0) = 60,0

=== ТЕСТИРОВАНИЕ ЗАВЕРШЕНО ===