

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 «Ознакомление со структурой Java»

Выполнил: Булавкина
Виктория Станиславовна

Группа: 6201-120303D

Самара, 2025

Задание №1

Я создала пакет `functions`. Этот пакет будет содержать все классы, созданные в результате выполнения данной программы.

Задание №2

Далее я разработала класс `FunctionPoint` и поместила его в пакет `functions`. Этот класс представляет собой точку табулированной функции. Класс реализован с соблюдением инкапсуляции и содержит два поля: координату X и координату Y . Для создания объектов доступны три конструктора:

С параметрами — для установки конкретных координат;

Копирующий — для создания дубликата существующей точки;

По умолчанию — инициализирует точку в начале координат $(0, 0)$.

```
1 package functions;
2
3 public class FunctionPoint { 16 usages viktoriabulavkina7-sys
4     public double x; 23 usages
5     public double y; 11 usages
6
7     // Конструктор с координатами
8     public FunctionPoint(double x, double y) { 24 usages viktoriabulavkina7-sys
9         this.x = x;
10        this.y = y;
11    }
12
13    // Конструктор копирования
14    @ public FunctionPoint(FunctionPoint point) { 22 usages viktoriabulavkina7-sys
15        this.x = point.x;
16        this.y = point.y;
17    }
18
19    // Конструктор по умолчанию
20    public FunctionPoint() { 20 usages viktoriabulavkina7-sys
21        this.x = 0;
22        this.y = 0;
23    }
24
25    @ public String toString() { viktoriabulavkina7-sys
26        return "(" + x + "; " + y + ")";
27    }
28 }
```

Задание №3

Далее в пакете functions я создала класс TabulatedFunction. Класс TabulatedFunction представляет табулированные функции. Данные хранятся в массиве точек FunctionPoint с обязательным упорядочиванием по X.

Конструкторы класса:

Создает равномерную сетку точек по заданному интервалу;

Распределяет массив значений равномерно по области определения.

```
package functions;

public class TabulatedFunction { 5 usages  viktoriabulavkina7-sys
    private FunctionPoint[] points; 34 usages
    private int pointsCount; 31 usages

    // Конструктор 1: по границам и количеству точек
    public TabulatedFunction(double leftX, double rightX, int pointsCount) { 17 usages  viktoriabulavkina7
        if (pointsCount < 2) {
            pointsCount = 2;
        }

        this.pointsCount = pointsCount;
        this.points = new FunctionPoint[pointsCount * 2]; // Запас места

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

        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, y: 0);
        }
    }

    // Конструктор 2: по границам и значениям
    public TabulatedFunction(double leftX, double rightX, double[] values) { 17 usages  viktoriabulavkina7
        this.pointsCount = values.length;
    }
}
```

```
// Конструктор 2: по границам и значениям
public TabulatedFunction(double leftX, double rightX, double[] values) { 17 usages viktoriabu
    this.pointsCount = values.length;
    if (pointsCount < 2) {
        pointsCount = 2;
    }

    this.points = new FunctionPoint[pointsCount * 2];

    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]);
    }
}
```

Задание №4

В классе TabulatedFunction я реализовала методы для работы с функцией:

getLeftDomainBorder() — X первой точки (левая граница);

getRightDomainBorder() — X последней точки (правая граница);

getFunctionValue(x) вычисляет значение в точке x, возвращая Double.NaN при x вне области определения.

Между точками используется линейная интерполяция.

```

41 // Границы области определения
42 public double getLeftDomainBorder() { 1 usage  @viktoriabulavkina7-sys
43     return points[0].x;
44 }
45
46 public double getRightDomainBorder() { 1 usage  @viktoriabulavkina7-sys
47     return points[pointsCount - 1].x;
48 }
49
50 // Значение функции с линейной интерполяцией
51 public double getFunctionValue(double x) { 3 usages  @viktoriabulavkina7-sys
52     if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
53         return Double.NaN; // Не число (Not a Number)
54     }
55
56     // Ищем интервал, содержащий x
57     for (int i = 0; i < pointsCount - 1; i++) {
58         double x1 = points[i].x;
59         double x2 = points[i + 1].x;
60
61         if (x >= x1 && x <= x2) {
62             // Линейная интерполяция:  $y = y_1 + (y_2 - y_1) * (x - x_1) / (x_2 - x_1)$ 
63             double y1 = points[i].y;
64             double y2 = points[i + 1].y;
65

```

```

61         if (x >= x1 && x <= x2) {
62             // Линейная интерполяция:  $y = y_1 + (y_2 - y_1) * (x - x_1) / (x_2 - x_1)$ 
63             double y1 = points[i].y;
64             double y2 = points[i + 1].y;
65
66             return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
67         }
68     }
69
70     return Double.NaN;
71 }
72

```

Задание №5

Я добавила методы в класс TabulatedFunction для работы с отдельными точками функции:

getPointsCount() — количество точек;

getPoint(index) — копия точки по индексу (защита данных);

setPoint(index, point) — замена точки с проверкой порядка по X;

getPointX(index) и getPointY(index) — чтение координат

setPointX(index, x) и setPointY(index, y) — изменение координат с валидацией.

```
// Методы для работы с точками
public int getPointsCount() { 1 usage  @viktoriabulavkina7-sys
    return pointsCount;
}

public FunctionPoint getPoint(int index) { no usages  @viktoriabulavkina7-sys
    if (index < 0 || index >= pointsCount) {
        return null;
    }
    return new FunctionPoint(points[index]);
}

public void setPoint(int index, FunctionPoint point) { 1 usage  @viktoriabulavkina7-sys
    if (index < 0 || index >= pointsCount) {
        return;
    }

    // Проверяем, что новая x находится между соседними точками
    if (index > 0 && point.x <= points[index - 1].x) {
        return;
    }
    if (index < pointsCount - 1 && point.x >= points[index + 1].x) {
        return;
    }
}
```

```

        points[index].x = x;
    }

    public double getPointY(int index) { no usages  viktoriabulavkina7-sys
        if (index < 0 || index >= pointsCount) {
            return Double.NaN;
        }
        return points[index].y;
    }

    public void setPointY(int index, double y) { 1 usage  viktoriabulavkina7-sys
        if (index < 0 || index >= pointsCount) {
            return;
        }
        points[index].y = y;
    }

    // Удаление точки
    public void deletePoint(int index) { 1 usage  viktoriabulavkina7-sys
        if (pointsCount <= 2 || index < 0 || index >= pointsCount) {
            return;
        }

        // Сдвигаем все элементы после index влево

```

```

        for (int i = index; i < pointsCount - 1; i++) {
            points[i] = points[i + 1];
        }

        pointsCount--;
        points[pointsCount] = null; // Очищаем последний элемент
    }

    // Добавление точки
    public void addPoint(FunctionPoint point) { 1 usage  viktoriabulavkina7-sys
        // Находим позицию для вставки (точки упорядочены по x)
        int insertIndex = 0;
        while (insertIndex < pointsCount && points[insertIndex].x < point.x) {
            insertIndex++;
        }

        // Если точка с такой x уже существует, не добавляем
        if (insertIndex < pointsCount && Math.abs(points[insertIndex].x - point.x) < 0.000001) {
            return;
        }

        // Проверяем, достаточно ли места в массиве
        if (pointsCount >= points.length) {
            // Увеличиваем массив в 2 раза

```

```

        FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
        for (int i = 0; i < pointsCount; i++) {
            newPoints[i] = points[i];
        }
        points = newPoints;
    }

    // Сдвигаем элементы вправо, начиная с insertIndex
    for (int i = pointsCount; i > insertIndex; i--) {
        points[i] = points[i - 1];
    }

    // Вставляем новую точку
    points[insertIndex] = new FunctionPoint(point);
    pointsCount++;
}

public String toString() {
    @ viktoriabulavkina7-sys
    StringBuilder result = new StringBuilder();
    result.append("TabulatedFunction ").append(pointsCount).append(" точек):\n");
    for (int i = 0; i < pointsCount; i++) {
        result.append(" ").append(i).append(": ").append(points[i]).append("\n");
    }
    return result.toString();
}

```

```

        return result.toString();
    }
}

```