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).

```java
package functions;

public class FunctionPoint {  16 usages  viktoriabulavkina7-sys
    public double x;  23 usages
    public double y;  11 usages

    // Конструктор с координатами
    public FunctionPoint(double x, double y) {  24 usages  viktoriabulavkina7-sys
        this.x = x;
        this.y = y;
    }

    // Конструктор копирования
    public FunctionPoint(FunctionPoint point) {  22 usages  viktoriabulavkina7-sys
        this.x = point.x;
        this.y = point.y;
    }

    // Конструктор по умолчанию
    public FunctionPoint() {  20 usages  viktoriabulavkina7-sys
        this.x = 0;
        this.y = 0;
    }

    public String toString() {  viktoriabulavkina7-sys
        return "(" + x + "; " + y + ")";
    }
}
```

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;
```

```java
// 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.

```
    // Границы области определения
    public double getLeftDomainBorder() {  1 usage   viktoriabulavkina7-sys
        return points[0].x;
    }

    public double getRightDomainBorder() {  1 usage   viktoriabulavkina7-sys
        return points[pointsCount - 1].x;
    }

    // Значение функции с линейной интерполяцией
    public double getFunctionValue(double x) {  3 usages   viktoriabulavkina7-sys
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN; // Не число (Not a Number)
        }

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

            if (x >= x1 && x <= x2) {
                // Линейная интерполяция: y = y1 + (y2-y1)*(x-x1)/(x2-x1)
                double y1 = points[i].y;
                double y2 = points[i + 1].y;
```

```
                if (x >= x1 && x <= x2) {
                    // Линейная интерполяция: y = y1 + (y2-y1)*(x-x1)/(x2-x1)
                    double y1 = points[i].y;
                    double y2 = points[i + 1].y;

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

            return Double.NaN;
        }
```

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();
        }
    }
```