МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Институт информатики и кибернетики
Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 2.

по курсу
Объектно-ориентированное программирование

Группа 6203-010302D
Студент _____ Д.А. Лихванова
(*подпись*)

Преподаватель,
к.ф.-м.н. _____ А.В. Благов
(*подпись*)

Самара 2025

Задание

Задание на лабораторную работу приведено в таблице 1.

Таблица 1 – Задание на ЛР № 2

Основное задание на ЛР	Разработать набор классов для работы с функциями одной переменной, заданными в табличной форме. Каждое следующее задание необходимо выполнить в новом каталоге.

Задание №1

В первом задании я создала структура проекта с папкой «functions» для будущих классов и файл «.gitignore».

Задание 2.

Далее я создала класс «FunctionPoint», где я создала два поля «private double x» и «private double y». Далее в соответствии с заданием в классе были созданы конструкторы: FunctionPoint(double x, double y), FunctionPoint(FunctionPoint point) и FunctionPoint(), также для удобства были написаны методы геттеры-сеттеры для полей данного класса.

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

    // Конструктор с параметрами
    public FunctionPoint(double x, double y) {
        this.x = x;
        this.y = y;
    }

    // Конструктор копирования
    public FunctionPoint(FunctionPoint point) {
        this.x = point.x;
        this.y = point.y;
    }

    // Конструктор по умолчанию
    public FunctionPoint() {
        this(x: 0.0, y: 0.0);
    }

    // Геттеры
    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

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

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

Задание 3.

В пакете «functions» разработан класс «TabulatedFunction» для работы с таблично заданными функциями. Класс содержит массив объектов «functionPoint» и счётчик реального количества точек.
Массив я создавала с запасом (pointsCount + 5) для оптимизации будущих операций добавления точек.
Реализованы два конструктора. Первый принимает границы области определения и количество точек, создавая точки с равными интервалами по X и нулевыми значениями Y. Второй конструктор получает массив значений Y, устанавливая их для равномерно распределённых точек.

Задание 4.

Здесь я реализовала методы для работы с областью определения и вычисления значений функции. Методы «getLeftDomainBorder()» и «getRightDomainBorder()» возвращают границы области определения функции.
Затем был реализован метод «getFunctionValue(double x)», который вычисляет значение функции в заданной точке. Вначале выполняется проверка принадлежности точки области определения функции. Если точка находится вне интервала, возвращается специальное значение «Double.NaN».

При реализации возникли трудности организации корректного поиска интервала для произвольной точки и обеспечение точности вычислений при линейной интерполяции.

Задание 5.

Для работы с отдельными точками табулированной функции были разработаны семь методов. Метод «getPointsCount()» возвращает текущее количество точек. Метод «getPoint(int index)» обеспечивает доступ к точке по индексу, создавая и возвращая её копию для соблюдения инкапсуляции — внешний код получает данные, но не может через возвращённый объект изменить внутреннее состояние массива. Для получения отдельных координат реализованы методы «getPointX(int index)» и «getPointY(int index)». Они используют соответствующие геттеры класса «FunctionPoint».

Методы модификации «setPoint(int index, FunctionPoint point)», «setPointX(int index, double x)» и «setPointY(int index, double y)» позволяют изменять точки. Ключевым требованием было сохранение свойства упорядоченности массива по координате X. Поэтому перед изменением абсциссы (setPoint и setPointX) выполняются проверки: для изменяемой точки, не являющейся крайней, новое значение X должно строго находиться между значениями X соседних точек. Для первой точки новое X должно быть меньше X второй точки, а для последней — больше X предпоследней. Эти проверки предотвращают нарушение порядка. Изменение ординаты (setPointY) таких проверок не требует.

Наиболее сложной оказалась реализация проверок в «setPointX» , где необходимо корректно обрабатывать три различных случая: изменение единственной точки, изменение крайних точек и изменение внутренней точки.

Задание 6.

В класс «TabulatedFunction» я добавила два метода для изменения количества точек функции.

Метод «deletePoint(int index)» выполняет удаление точки по указанному индексу. Метод включает проверку на минимальное количество точек — табулированная функция не может содержать менее двух точек. Перед удалением проверяется корректность индекса. Для удаления точки используется алгоритм сдвига элементов массива влево, начиная с позиции после удаляемого элемента. Последний элемент массива обнуляется.

Метод «addPoint(FunctionPoint point)» добавляет новую точку в функцию с сохранением упорядоченности массива по координате X. Если внутренний массив заполнен, он динамически расширяется. Алгоритм находит правильную позицию для вставки, проверяя, что точка с таким значением X ещё не существует. Затем происходит сдвиг элементов вправо для освобождения места, и в освободившуюся позицию помещается копия переданной точки.

Задание 7.

Для проверки работоспособности реализованных классов был создан класс Main, расположенный вне пакета functions. В методе main() выполнялось тестирование функциональности табулированной функции. Была создана табулированная функция $y = x^2$ на интервале [0, 4] с 5 точками.

В методе «addPoint(FunctionPoint point)» я проверила вставку новой точки в середину массива с сохранением упорядоченности по X. Также протестировала случай добавления точки с уже существующим X - в этом случае метод обновляет значение Y существующей точки.

В методе «deletePoint(int index)» также проверила корректное удаление точки из середины массива. После удаления остальные точки сдвигаются, а последняя ячейка массива обнуляется. Метод предотвращает удаление, если останется менее 2 точек.

```
>> java Main
=== TESTING POINT METHODS ===ava\x0ajavac Main.java\x0a\x0a# 3.

1. CREATING INITIAL FUNCTION y = x^2:
   Initial function points:
   [0] (0,00; 0,00)
   [1] (1,00; 1,00)
   [2] (2,00; 4,00)
   [3] (3,00; 9,00)
   [4] (4,00; 16,00)

2. TEST addPoint():
   Adding point (2.5; 6.25) in the middle:
   Expected: point inserted between (2.0; 4.0) and (3.0; 9.0)
   After addPoint(2.5; 6.25):
   [0] (0,00; 0,00)
   [1] (1,00; 1,00)
   [2] (2,00; 4,00)
   [3] (2,50; 6,25)
   [4] (3,00; 9,00)
   [5] (4,00; 16,00)

3. TEST addPoint() WITH EXISTING X (update Y):
   Adding point (2.5; 100.0):
   Expected: Y of point (2.5; 6.25) changes to 100.0
   After addPoint(2.5; 100.0):
   [0] (0,00; 0,00)
   [1] (1,00; 1,00)
   [2] (2,00; 4,00)
   [3] (2,50; 6,25)
   [4] (3,00; 9,00)
   [5] (4,00; 16,00)
```

```
4. TEST deletePoint():
   Deleting point at index 2 (value (2.0; 4.0)):
   Expected: point removed, others shifted
   After deletePoint(2):
   [0] (0,00; 0,00)
   [1] (1,00; 1,00)
   [2] (2,50; 6,25)
   [3] (3,00; 9,00)
   [4] (4,00; 16,00)

5. TEST setPoint() - CORRECT REPLACEMENT:
   Replacing point at index 1 (1.0; 1.0) with (1.5; 2.25):
   Expected: point changed, order preserved
   Before: point [1] = (1,00; 1,00)
   After: point [1] = (1,50; 2,25)
   Final state after setPoint(1, (1.5; 2.25)):
   [0] (0,00; 0,00)
   [1] (1,50; 2,25)
   Expected: point removed, others shifted
   After deletePoint(2):
   [0] (0,00; 0,00)
   [1] (1,00; 1,00)
   [2] (2,50; 6,25)
   [3] (3,00; 9,00)
   [4] (4,00; 16,00)
```

```
6. TEST setPoint() - INCORRECT REPLACEMENT:
   Trying to replace point [1] with (3.0; 9.0):
   Expected: DOES NOT CHANGE (X=3.0 violates order)
   Before replacement: point [1] = (1,50; 2,25)
   After attempt: point [1] = (1,50; 2,25)
```

```
=== FINAL FUNCTION STATE ===
   Function after all operations:
   Function after all operations:
   [0] (0,00; 0,00)
   [0] (0,00; 0,00)
   [1] (1,50; 2,25)
   [2] (2,50; 6,25)
   [3] (3,00; 9,00)
   [4] (4,00; 16,00)

=== ALL TESTS COMPLETED ===
PS C:\Users\MOON\Lab2>
```