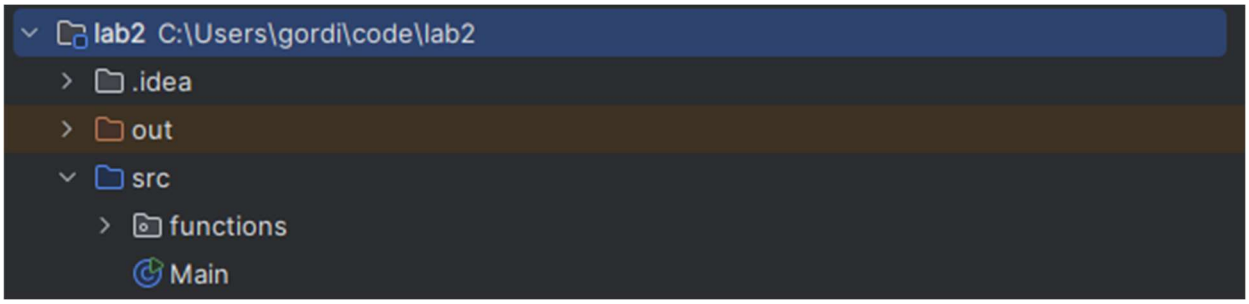
Лабораторная работа №2.

Выполнил: Горынин Дмитрий, ст. гр. 6201-120303D.

Задание 1.

Создать отдельный пакет для классов, используемых в лабораторной работе.



Задание 2.

Создать в пакете functions класс FunctionPoint, который будет описывать точку функции.

```java
package functions;

public class FunctionPoint {  25 usages
    private double x;  5 usages
    private double y;  5 usages

    // Конструктор с заданными координатами
    public FunctionPoint(double x, double y) {  8 usages
        this.x = x;
        this.y = y;
    }

    // Копирующий конструктор
    public FunctionPoint(FunctionPoint point) {  3 usages
        this.x = point.x;
        this.y = point.y;
    }

    // Конструктор по умолчанию
    public FunctionPoint() {  no usages
        this( x: 0.0,  y: 0.0);
    }

    // Геттеры и сеттеры с инкапсуляцией
    public double getX() {  17 usages
        return x;
    }

    public void setX(double x) {  1 usage
        this.x = x;
    }

    public double getY() {  6 usages
        return y;
    }

    public void setY(double y) {  1 usage
        this.y = y;
    }

}
```

Задание 3.

В пакете functions создать класс TabulatedFunction, который будет описывать табулированную функцию.

```
package functions;

public class TabulatedFunction {  3 usages
    private FunctionPoint[] points;  39 usages
    private int size; // Текущее количество точек  27 usages

    // Конструкторы
    public TabulatedFunction(double leftX, double rightX, int pointsCount) {  no usages
        if (pointsCount < 2) {
            throw new IllegalArgumentException("pointsCount must be at least 2");
        }
        if (leftX >= rightX) {
            throw new IllegalArgumentException("leftX must be less than rightX");
        }

        this.points = new FunctionPoint[pointsCount + 2]; // Запас памяти
        this.size = pointsCount;
        double step = (rightX - leftX) / (pointsCount - 1);

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

```
public TabulatedFunction(double leftX, double rightX, double[] values) {  1 usage
    if (values.length < 2) {
        throw new IllegalArgumentException("values array must have at least 2 elements");
    }
    if (leftX >= rightX) {
        throw new IllegalArgumentException("leftX must be less than rightX");
    }

    int pointsCount = values.length;
    this.points = new FunctionPoint[pointsCount + 2]; // Запас памяти
    this.size = pointsCount;
    double step = (rightX - leftX) / (pointsCount - 1);

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

Задание 4.

В классе TabulatedFunction описать методы, необходимые для работы с функцией.

```java
public double getLeftDomainBorder() {  2 usages
    return points[0].getX();
}

public double getRightDomainBorder() {  2 usages
    return points[size - 1].getX();
}

public double getFunctionValue(double x) {  2 usages
    if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
        return Double.NaN;
    }

    for (int i = 0; i < size - 1; i++) {
        double x1 = points[i].getX();
        double x2 = points[i + 1].getX();

        if (x == x1) return points[i].getY();
        if (x == x2) return points[i + 1].getY();

        if (x > x1 && x < x2) {
            return linearInterpolation(points[i], points[i + 1], x);
        }
    }

    return Double.NaN;
}
```

Задание 5.

В классе TabulatedFunction описать методы, необходимые для работы с точками табулированной функции.

```java
public int getPointsCount() {  6 usages
    return size;
}

public FunctionPoint getPoint(int index) {  6 usages
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }
    return new FunctionPoint(points[index]);
}

public void setPoint(int index, FunctionPoint point) {  no usages
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }

    if (!isValidXPosition(index, point.getX())) {
        throw new IllegalArgumentException(
                "New x-coordinate " + point.getX() + " at index " + index +
                        " would violate the ordering of points"
        );
    }

    points[index] = new FunctionPoint(point);
}
```

```java
public double getPointX(int index) {  no usages
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }
    return points[index].getX();
}

public void setPointX(int index, double x) {  1 usage
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }

    if (!isValidXPosition(index, x)) {
        throw new IllegalArgumentException(
                "New x-coordinate " + x + " at index " + index +
                        " would violate the ordering of points"
        );
    }

    double y = points[index].getY();
    points[index] = new FunctionPoint(x, y);
}
```

```java
public double getPointY(int index) {  no usages
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }
    return points[index].getY();
}

public void setPointY(int index, double y) {  1 usage
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }

    double x = points[index].getX();
    points[index] = new FunctionPoint(x, y);
}
```

Задание 6.

В классе TabulatedFunction описать методы, изменяющие количество точек табулированной функции.

```java
public void deletePoint(int index) {  3 usages
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }
    if (size <= 2) {
        throw new IllegalStateException("Cannot delete point - function must have at least 2 points");
    }

    // Сдвигаем точки влево, начиная с позиции после удаляемой
    System.arraycopy(points,  srcPos: index + 1, points, index,  length: size - index - 1);
    points[size - 1] = null; // Помогаем сборщику мусора
    size--;

    // Если массив слишком пустой, уменьшаем его размер (но не меньше начального + 2)
    if (points.length > size * 2 && points.length > 4) {
        shrinkArray();
    }
}
```

```java
public void addPoint(FunctionPoint point) {  3 usages
    // Находим позицию для вставки (сохраняя упорядоченность по x)
    int insertIndex = 0;
    while (insertIndex < size && points[insertIndex].getX() < point.getX()) {
        insertIndex++;
    }

    // Проверяем, что точка с таким x уже не существует
    if (insertIndex < size && Math.abs(points[insertIndex].getX() - point.getX()) < 1e-10) {
        throw new IllegalArgumentException("Point with x=" + point.getX() + " already exists");
    }

    // Проверяем, нужно ли увеличивать массив
    if (size >= points.length) {
        expandArray();
    }

    // Сдвигаем точки вправо, чтобы освободить место для новой точки
    if (insertIndex < size) {
        System.arraycopy(points, insertIndex, points,  destPos: insertIndex + 1,  length: size - insertIndex);
    }

    // Вставляем копию точки (инкапсуляция)
    points[insertIndex] = new FunctionPoint(point);
    size++;
}
```

Задание 7.

Проверить работу написанных классов.

```java
import functions.FunctionPoint;
import functions.TabulatedFunction;

public class Main {
    public static void main(String[] args) {
        System.out.println("=== Тестирование класса TabulatedFunction ===\n");

        // 1. Создаем табулированную функцию для y = x^2 на интервале [0, 4]
        System.out.println("1. Создание функции y = x^2 на [0, 4] с 5 точками:");
        double[] values = {0, 1, 4, 9, 16};
        TabulatedFunction func = new TabulatedFunction( leftX: 0,  rightX: 4, values);
        System.out.println("Функция: " + func);
        System.out.println("Количество точек: " + func.getPointsCount());
        System.out.println("Область определения: [" + func.getLeftDomainBorder() + ", " + func.getRightDomainBorder() + "]");

        // 2. Вычисляем значения функции в различных точках
        System.out.println("\n2. Вычисление значений функции:");
        double[] testPoints = {-1, 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 5};

        for (double x : testPoints) {
            double y = func.getFunctionValue(x);
            if (Double.isNaN(y)) {
                System.out.printf("f(%.1f) = вне области определения%n", x);
            } else {
                System.out.printf("f(%.1f) = %.2f%n", x, y);
            }
        }

        // 3. Тестируем модификацию точек
        System.out.println("\n3. Модификация точек:");

        // Изменяем значение y в точке x=2
        System.out.println("До изменения: " + func.getPoint( index: 2));
        func.setPointY( index: 2,  y: 5.0);
        System.out.println("После setPointY(2, 5.0): " + func.getPoint( index: 2));

        // Пытаемся изменить x точки (должно сохранить упорядоченность)
        try {
            func.setPointX( index: 2,  x: 1.5);
            System.out.println("После setPointX(2, 1.5): " + func.getPoint( index: 2));
        } catch (IllegalArgumentException e) {
            System.out.println("Ошибка при setPointX: " + e.getMessage());
        }

        // 4. Тестируем добавление точек
        System.out.println("\n4. Добавление точек:");

        // Добавляем точку в середину
        func.addPoint(new FunctionPoint( x: 0.7,  y: 0.49));
        System.out.println("После добавления (0.7, 0.49):");
        System.out.println("Количество точек: " + func.getPointsCount());
        System.out.println("Функция: " + func);

        // Добавляем точку в начало
        func.addPoint(new FunctionPoint( x: -0.5,  y: 0.25));
        System.out.println("После добавления (-0.5, 0.25):");
        System.out.println("Количество точек: " + func.getPointsCount());
        System.out.println("Функция: " + func);

        // 5. Тестируем удаление точек
        System.out.println("\n5. Удаление точек:");

        // Удаляем точку с индексом 2
        System.out.println("Удаляем точку с индексом 2: " + func.getPoint( index: 2));
        func.deletePoint( index: 2);
        System.out.println("После удаления:");
        System.out.println("Количество точек: " + func.getPointsCount());
        System.out.println("Функция: " + func);

        // 6. Проверяем значения после изменений
        System.out.println("\n6. Проверка значений после изменений:");
        double[] newTestPoints = {-0.5, 0, 0.7, 1, 2, 3, 4};
```

```
    for (double x : newTestPoints) {
        double y = func.getFunctionValue(x);
        if (Double.isNaN(y)) {
            System.out.printf("f(%.1f) = вне области определения%n", x);
        } else {
            System.out.printf("f(%.1f) = %.2f%n", x, y);
        }
    }

    // 7. Тестируем граничные случаи
    System.out.println("\n7. Граничные случаи:");

    // Попытка добавить точку с существующим x
    try {
        func.addPoint(new FunctionPoint( x: 1.0,  y: 100));
        System.out.println("Точка добавлена");
    } catch (IllegalArgumentException e) {
        System.out.println("Ошибка при добавлении точки с существующим x: " + e.getMessage());
    }

    // Попытка удалить точку, когда останется только 1 точка
    try {
        // Удаляем точки пока не останется 2
        while (func.getPointsCount() > 2) {
            func.deletePoint( index: 0);
        }
        System.out.println("Осталось точек: " + func.getPointsCount());

        // Пытаемся удалить еще одну (должна быть ошибка)
        func.deletePoint( index: 0);
    } catch (IllegalStateException e) {
        System.out.println("Ошибка при удалении последней точки: " + e.getMessage());
    }
```

```
        // 8. Проверка инкапсуляции
        System.out.println("\n8. Проверка инкапсуляции:");
        FunctionPoint testPoint = func.getPoint( index: 0);
        System.out.println("Полученная точка: " + testPoint);

        // Меняем полученную точку (не должна влиять на оригинал)
        testPoint.setX(999);
        testPoint.setY(999);
        System.out.println("После изменения полученной точки:");
        System.out.println("Полученная точка: " + testPoint);
        System.out.println("Оригинальная точка в функции: " + func.getPoint( index: 0));

        System.out.println("\n=== Тестирование завершено ===");
    }
}
```

Результат:

```
1. Создание функции y = x^2 на [0, 4] с 5 точками:
Функция: functions.TabulatedFunction@30f39991
Количество точек: 5
Область определения: [0.0, 4.0]

2. Вычисление значений функции:
f(-1,0) = вне области определения
f(0,0) = 0,00
f(0,5) = 0,50
f(1,0) = 1,00
f(1,5) = 2,50
f(2,0) = 4,00
f(2,5) = 6,50
f(3,0) = 9,00
f(3,5) = 12,50
f(4,0) = 16,00
f(5,0) = вне области определения

3. Модификация точек:
До изменения: functions.FunctionPoint@3af49f1c
После setPointY(2, 5.0): functions.FunctionPoint@19469ea2
После setPointX(2, 1.5): functions.FunctionPoint@13221655
```

```
4. Добавление точек:
После добавления (0.7, 0.49):
Количество точек: 6
Функция: functions.TabulatedFunction@30f39991
После добавления (-0.5, 0.25):
Количество точек: 7
Функция: functions.TabulatedFunction@30f39991

5. Удаление точек:
Удаляем точку с индексом 2: functions.FunctionPoint@2f2c9b19
После удаления:
Количество точек: 6
Функция: functions.TabulatedFunction@30f39991

6. Проверка значений после изменений:
f(-0,5) = 0,25
f(0,0) = 0,00
f(0,7) = 0,70
f(1,0) = 1,00
f(2,0) = 6,33
f(3,0) = 9,00
f(4,0) = 16,00
```

```
7. Граничные случаи:
Ошибка при добавлении точки с существующим x: Point with x=1.0 already exists
Осталось точек: 2
Ошибка при удалении последней точки: Cannot delete point - function must have at least 2 points

8. Проверка инкапсуляции:
Полученная точка: functions.FunctionPoint@1fb3ebeb
После изменения полученной точки:
Полученная точка: functions.FunctionPoint@1fb3ebeb
Оригинальная точка в функции: functions.FunctionPoint@548c4f57
```