ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА»

Институт «Информатики и кибернетики»

Специальность «Фотоника и оптоинформатика 6201-120303D»

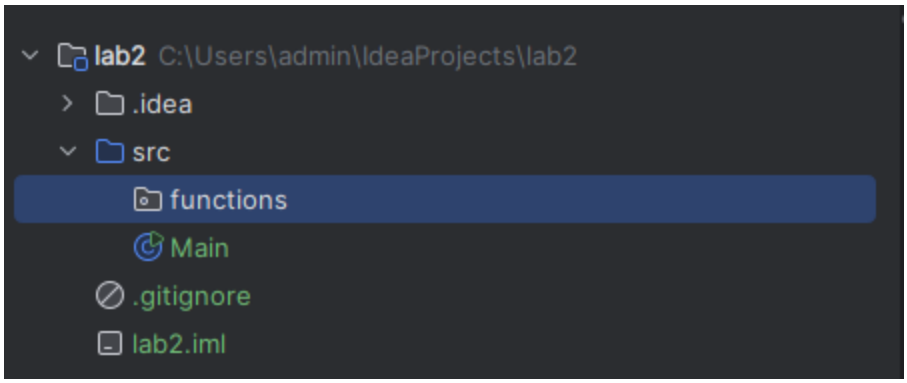
Отчет по лабораторной работе № 2

Выполнил: студент Султанова А.М.,
группа 6201-120303D
Проверил: преподаватель Борисов Д.С.

Самара 2025

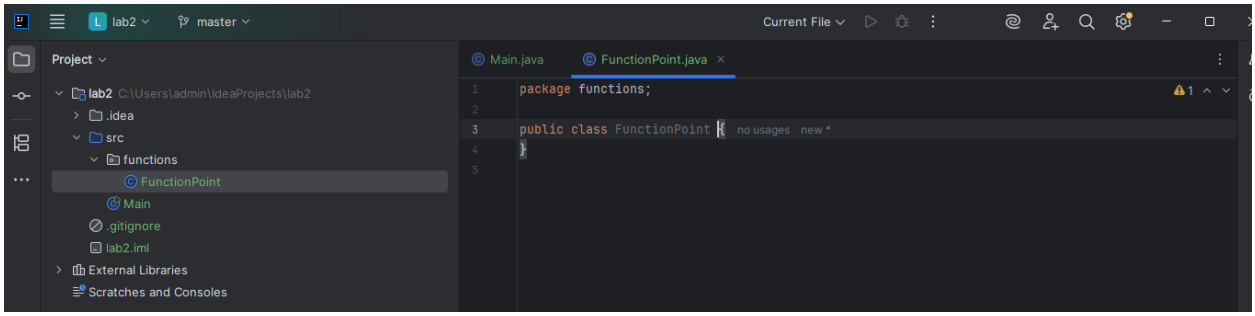
Задание 1

Создаем пакет functions, в котором далее будут создаваться классы программы.

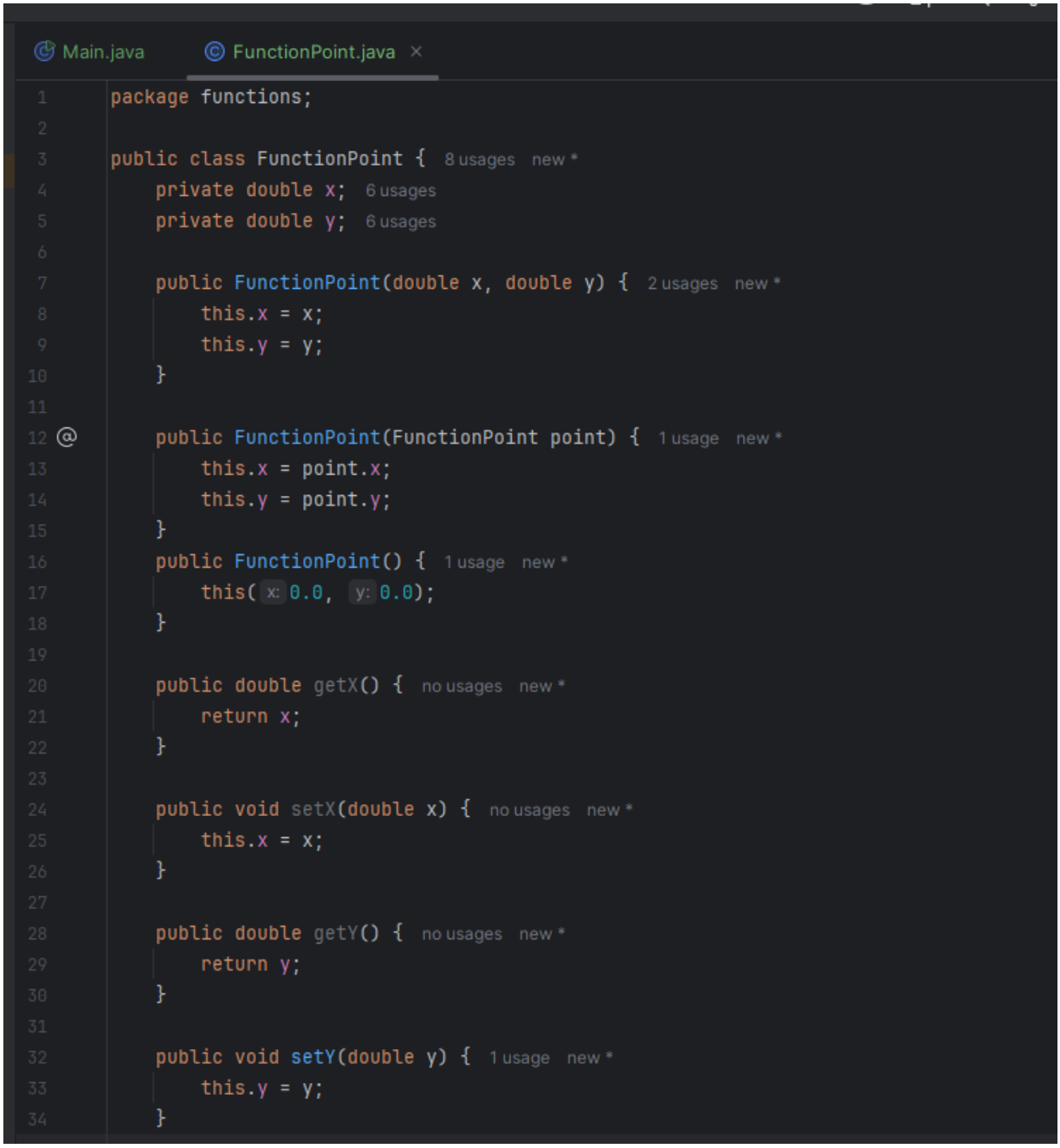


Задание 2

В пакете functions создаем класс FunctionPoint



Пишем класс FunctionPoint, который описывает одну точку табулированной функции, храня координаты x и y, предоставляя методы для работы с ними.



Задание 3

В пакете functions создаем класс TabulatedFunction

Пишем класс TabulatedFunction, который хранит и создаёт табулированную функцию в виде массива точек FunctionPoint, равномерно распределённых по оси x с заданными значениями y.

Main.java | FunctionPoint.java | TabulatedFunction.java

```java
package functions;

public class TabulatedFunction {  3 usages  new *
    private FunctionPoint[] points;  25 usages
    private int size;  13 usages

    public TabulatedFunction(double leftX, double rightX, int pointsCount) {  1 usage  new *
        this.points = new FunctionPoint[pointsCount];
        this.size = pointsCount;
        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x,  y: 0.0);
        }
    }

    public TabulatedFunction(double leftX, double rightX, double[] values) {  no usages  new *
        int n = values.length;
        this.points = new FunctionPoint[n];
        this.size = n;
        double step = (rightX - leftX) / (n - 1);
        for (int i = 0; i < n; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, values[i]);
        }
    }
```

Задание 4

Добавим к коду следующие методы:

Метод getLeftDomainBorder() возвращает минимальный x.

Метод getRightDomainBorder() возвращает максимальный x.

Метод getFunctionValue(x) возвращает точное значение, если x совпадает с точкой, либо интерполированное значение, если x внутри области, или

Double.NaN, если x вне области.

```
27
28      public double getLeftDomainBorder() {  no usages  new *
29          return points[0].getX();
30      }
31
32      public double getRightDomainBorder() {  no usages  new *
33          return points[size - 1].getX();
34      }
35
36      public double getFunctionValue(double x) {  1 usage  new *
37          for (int i = 0; i < size - 1; i++) {
38              double x0 = points[i].getX();
39              double x1 = points[i + 1].getX();
40              double y0 = points[i].getY();
41              double y1 = points[i + 1].getY();
42
43              if (x == x0) return y0;
44              if (x == x1) return y1;
45
46              if (x > x0 && x < x1) {
47                  return y0 + (y1 - y0) * (x - x0) / (x1 - x0);
48              }
49          }
50          return Double.NaN;
51      }
```

Задание 5

Добавим следующие методы:

getPointsCount() — возвращает количество точек функции.

getPoint(int index) — возвращает копию точки по индексу.

setPoint(int index, FunctionPoint point) — заменяет точку новой, если новая не нарушает порядок по x.

getPointX(int index) — возвращает значение x точки по индексу.

setPointX(int index, double x) — изменяет x точки, если остаётся между соседями.

getPointY(int index) — возвращает значение Y точки по индексу.

setPointY(int index, double y) — изменяет Y точки.

```
    public int getPointsCount() {  3 usages  new *
        return size;
    }

    public FunctionPoint getPoint(int index) {  no usages  new *
        return new FunctionPoint(points[index]);
    }

    public void setPoint(int index, FunctionPoint point) {  no usages  new *
        points[index] = new FunctionPoint(point);
    }

    public double getPointX(int index) {  4 usages  new *
        return points[index].getX();
    }

    public void setPointX(int index, double x) {  no usages  new *
        points[index].setX(x);
    }

    public double getPointY(int index) {  3 usages  new *
        return points[index].getY();
    }

    public void setPointY(int index, double y) {  2 usages  new *
        points[index].setY(y);
    }
```

Задание 6

Добавим к коду следующие методы:

deletePoint(int index) — удаляет точку по индексу, сдвигая оставшиеся влево.

addPoint(FunctionPoint point) — добавляет новую точку в нужное место, сохраняя порядок по x.

```
80
81      public void deletePoint(int index) {  1 usage  new *
82          for (int i = index; i < size - 1; i++) {
83              points[i] = points[i + 1];
84          }
85          size--;
86      }
87
88      public void addPoint(FunctionPoint point) {  1 usage  new *
89          if (size == points.length) {
90              FunctionPoint[] newPoints = new FunctionPoint[size * 2];
91              System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, size);
92              points = newPoints;
93          }
94
95          int insertIndex = 0;
96          double x = point.getX();
97          while (insertIndex < size && points[insertIndex].getX() < x) {
98              insertIndex++;
99          }
100
101         for (int i = size; i > insertIndex; i--) {
102             points[i] = points[i - 1];
103         }
104
105         points[insertIndex] = new FunctionPoint(point);
106         size++;
107     }
108 }
109
```

Задание 7

Проверяем работу написанных классов.

main() — запускает программу и проверяет работу всех методов.

new TabulatedFunction(left, right, count) — создаёт табулированную функцию на заданном отрезке.

Цикл с setPointY(i, x * x * x) — задаёт значения функции $y = x^3$.

Цикл с getFunctionValue(x) — выводит значения функции для разных x, включая точки вне области.

setPointY(1, 10) — изменяет значение ординаты одной из точек.

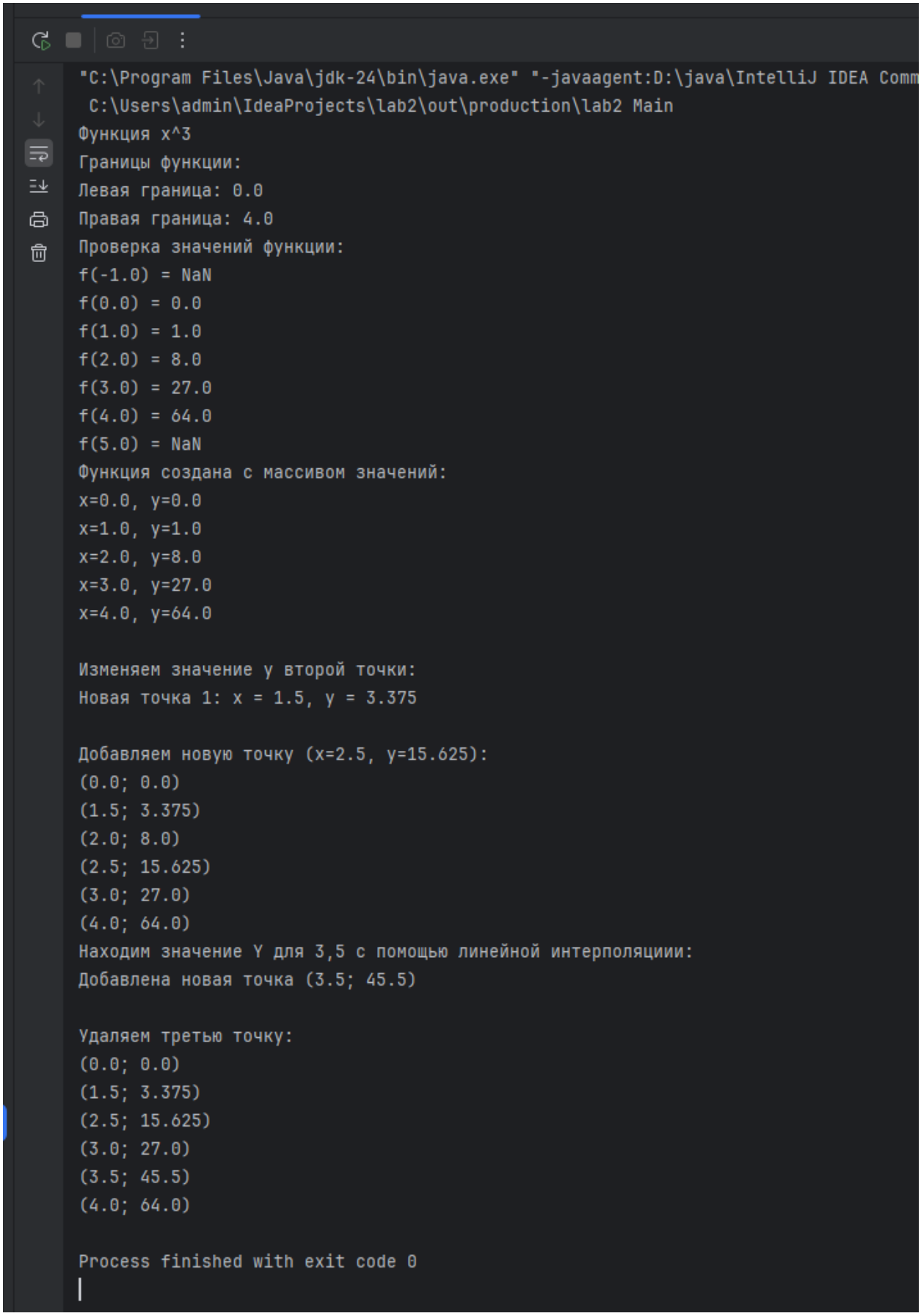
addPoint(new FunctionPoint(2.5, 15.625)) — добавляет новую точку в таблицу.

deletePoint(2) — удаляет точку по индексу.

Main.java × FunctionPoint.java TabulatedFunction.java

```java

import functions.FunctionPoint;
import functions.TabulatedFunction;

public class Main {
    public static void main(String[] args) {
        double left = 0;
        double right = 4;
        int count = 5;
        TabulatedFunction func = new TabulatedFunction(left, right, count);
        System.out.println("Функция x^3");
        System.out.println("Границы функции:");
        System.out.println("Левая граница: " + func.getLeftDomainBorder());
        System.out.println("Правая граница: " + func.getRightDomainBorder());
        for (int i = 0; i < func.getPointsCount(); i++) {
            double x = func.getPointX(i);
            func.setPointY(i, x * x * x);
        }

        System.out.println("Проверка значений функции:");
        for (double x = -1; x <= 5; x += 1.0) {
            System.out.println("f(" + x + ") = " + func.getFunctionValue(x));
        }
        double[] values = {0, 1, 8, 27, 64};
        TabulatedFunction func1 = new TabulatedFunction(left, right, values);

        System.out.println("Функция создана с массивом значений:");
        for (int i = 0; i < func1.getPointsCount(); i++) {
            System.out.println("x=" + func1.getPointX(i) + ", y=" + func1.getPointY(i));
        }

        System.out.println("\nИзменяем значение y второй точки:");
        func.setPoint( index: 1, new FunctionPoint( x: 1.5,  y: 3.375));
        System.out.println("Новая точка 1: x = " + func.getPointX( index: 1) + ", y = " + func.getPointY( index: 1));

        System.out.println("\nДобавляем новую точку (x=2.5, y=15.625):");
        func.addPoint(new FunctionPoint( x: 2.5,  y: 15.625));
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("(" + func.getPointX(i) + "; " + func.getPointY(i) + ")");
        }
        System.out.println("Находим значение Y для 3,5 с помощью линейной интерполяциии:");
        func.addPoint(new FunctionPoint( x: 3.5, func.getFunctionValue( x: 3.5))); // добавляем как новую точку
        System.out.println("Добавлена новая точка (3.5; " +func.getFunctionValue( x: 3.5) + ")");

        System.out.println("\nУдаляем третью точку:");
        func.deletePoint( index: 2);
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("(" + func.getPointX(i) + "; " + func.getPointY(i) + ")");
        }
    }
}

```

Проверяем:


"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:D:\java\IntelliJ IDEA Comm
C:\Users\admin\IdeaProjects\lab2\out\production\lab2 Main
Функция x^3
Границы функции:
Левая граница: 0.0
Правая граница: 4.0
Проверка значений функции:
f(-1.0) = NaN
f(0.0) = 0.0
f(1.0) = 1.0
f(2.0) = 8.0
f(3.0) = 27.0
f(4.0) = 64.0
f(5.0) = NaN
Функция создана с массивом значений:
x=0.0, y=0.0
x=1.0, y=1.0
x=2.0, y=8.0
x=3.0, y=27.0
x=4.0, y=64.0
Изменяем значение y второй точки:
Новая точка 1: x = 1.5, y = 3.375
Добавляем новую точку (x=2.5, y=15.625):
(0.0; 0.0)
(1.5; 3.375)
(2.0; 8.0)
(2.5; 15.625)
(3.0; 27.0)
(4.0; 64.0)
Находим значение Y для 3,5 с помощью линейной интерполяциии:
Добавлена новая точка (3.5; 45.5)
Удаляем третью точку:
(0.0; 0.0)
(1.5; 3.375)
(2.5; 15.625)
(3.0; 27.0)
(3.5; 45.5)
(4.0; 64.0)
Process finished with exit code 0