ЛАБОРАТОРНАЯ РАБОТА №4

Курс: Объектно-ориентированное программирование.

Студент: Кузнецов Эрик Витальевич
Группа:6204-010302D
Преподаватель: Борисов Дмитрий Сергеевич

Задание 1

Ход выполнения работы:

1)В класс ArrayTabulatedFunction добавляем конструктор, принимающий массив точек FunctionPoint[]

2)Проверяем, что количество точек не менее 2, иначе выбрасываем IllegalArgumentException

3)Проверяем упорядоченность точек по значению X, иначе выбрасываем IllegalArgumentException

4)Создаем копии точек для обеспечения инкапсуляции

5)В класс LinkedListTabulatedFunction добавляем аналогичный конструктор.

```
    }

    // ЗАДАНИЕ 1
    public ArrayTabulatedFunction(FunctionPoint[] points) {
        if (points.length < 2) {
            throw new IllegalArgumentException("Количество точек должно быть не менее 2");
        }

        // Проверяем упорядоченность точек
        for (int i = 1; i < points.length; i++) {
            if (points[i].getX() <= points[i-1].getX()) {
                throw new IllegalArgumentException("Точки не упорядочены по возрастанию x");
            }
        }

        this.pointsCount = points.length;
        this.points = new FunctionPoint[pointsCount + 10];
        for (int i = 0; i < pointsCount; i++) {
            this.points[i] = new FunctionPoint(points[i]);
        }
    }
```

```
public LinkedListTabulatedFunction(FunctionPoint[] points) {//новый конструктор (задание 1)
    this();
    if (points.length < 2) {
        throw new IllegalArgumentException("Количество точек должно быть не менее 2");
    }

    // Проверяем упорядоченность точек
    for (int i = 1; i < points.length; i++) {
        if (points[i].getX() <= points[i-1].getX()) {
            throw new IllegalArgumentException("Точки не упорядочены по возрастанию x");
        }
    }

    for (FunctionPoint point : points) {
        addNodeToTail().point = new FunctionPoint(point);
    }
}
```

Рисунок 1,2-изображение с конечным результатом для задания 1.

Задание 2.

Ход выполнения работы:

1)Создаем интерфейс Function в пакете functions

2)Добавляем методы в интерфейс:

getLeftDomainBorder() - возвращает левую границу области определения

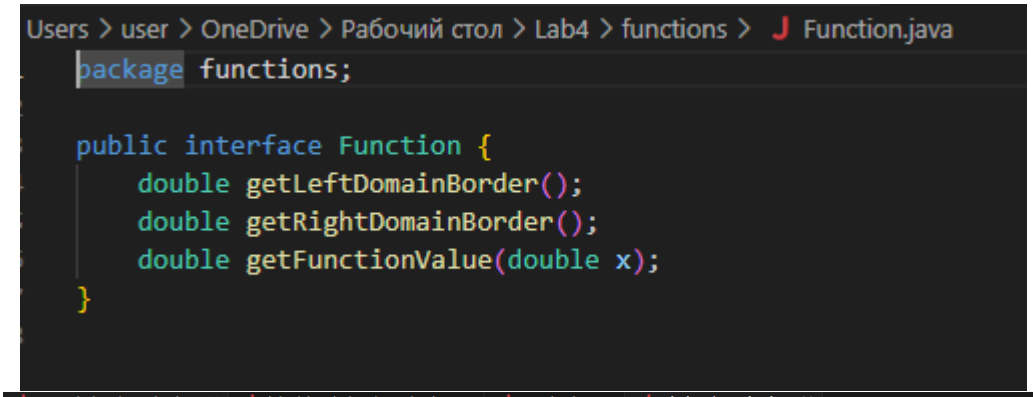
getRightDomainBorder() - возвращает правую границу области определения

getFunctionValue(double x) - возвращает значение функции в точке

3)Убираем эти методы из интерфейса TabulatedFunction

4)Делаем так, чтобы TabulatedFunction расширял интерфейс Function

5)Реализуем эти методы в классах ArrayTabulatedFunction и LinkedListTabulatedFunction



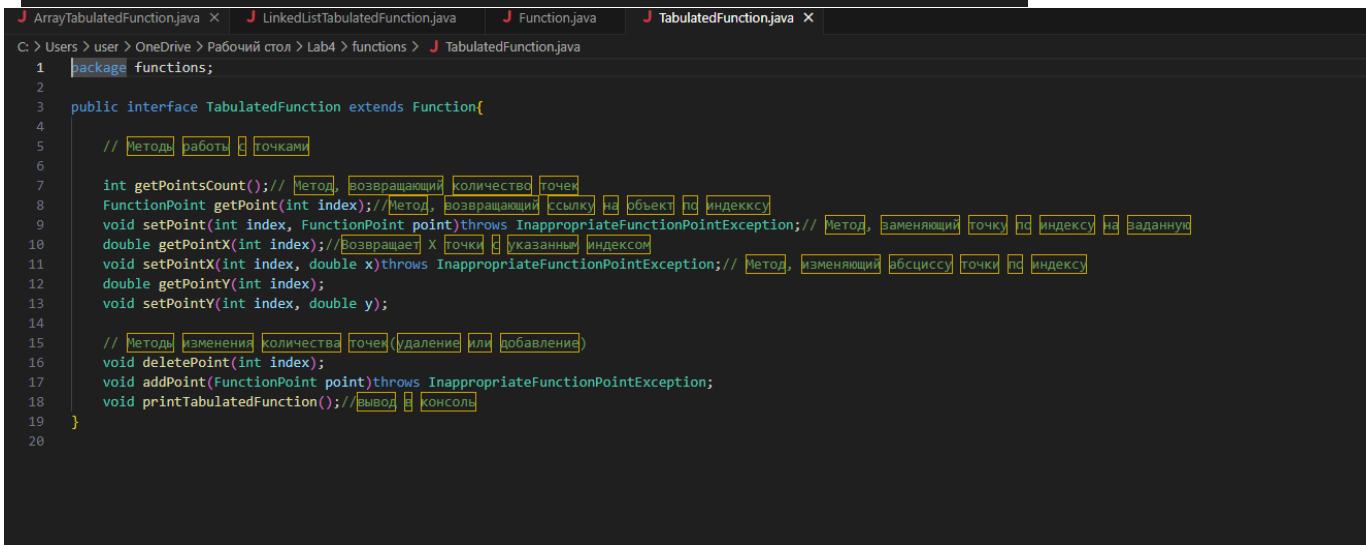


Рисунок 3,4-Снимок Экрана с конечным результатом для задания 2

Задание 3.

Ход выполнения работы:

1)Создаем пакет functions.basic для аналитических функций

2)Создаем класс Exp, реализующий интерфейс Function:

getLeftDomainBorder() возвращает Double.NEGATIVE_INFINITY

getRightDomainBorder() возвращает Double.POSITIVE_INFINITY

getFunctionValue() использует Math.exp()

3)Создаем класс Log, реализующий интерфейс Function:

Конструктор принимает основание логарифма

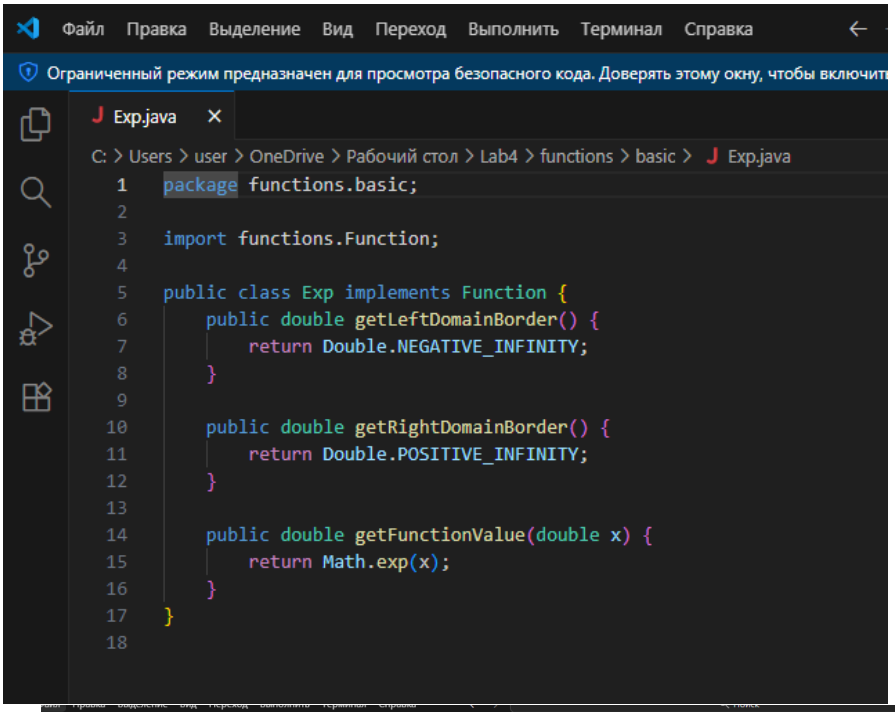
Проверяем основание > 0 и ≠ 1

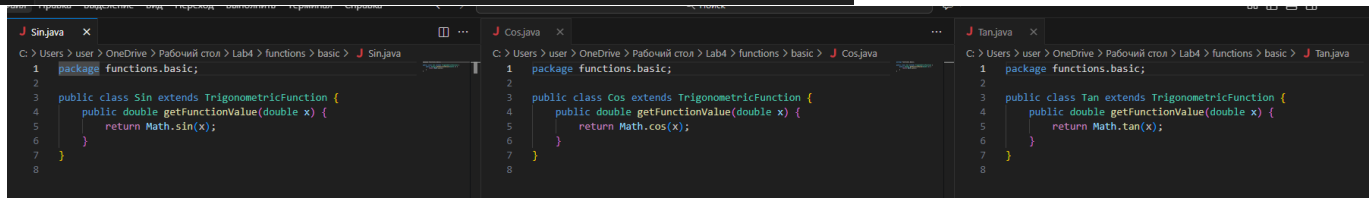
getFunctionValue() использует Math.log()

4)Создаем абстрактный класс TrigonometricFunction с реализацией методов границ области определения

5)Создаем классы Sin, Cos, Tan, наследующие от TrigonometricFunction:

getFunctionValue() использует Math.sin(), Math.cos(), Math.tan()





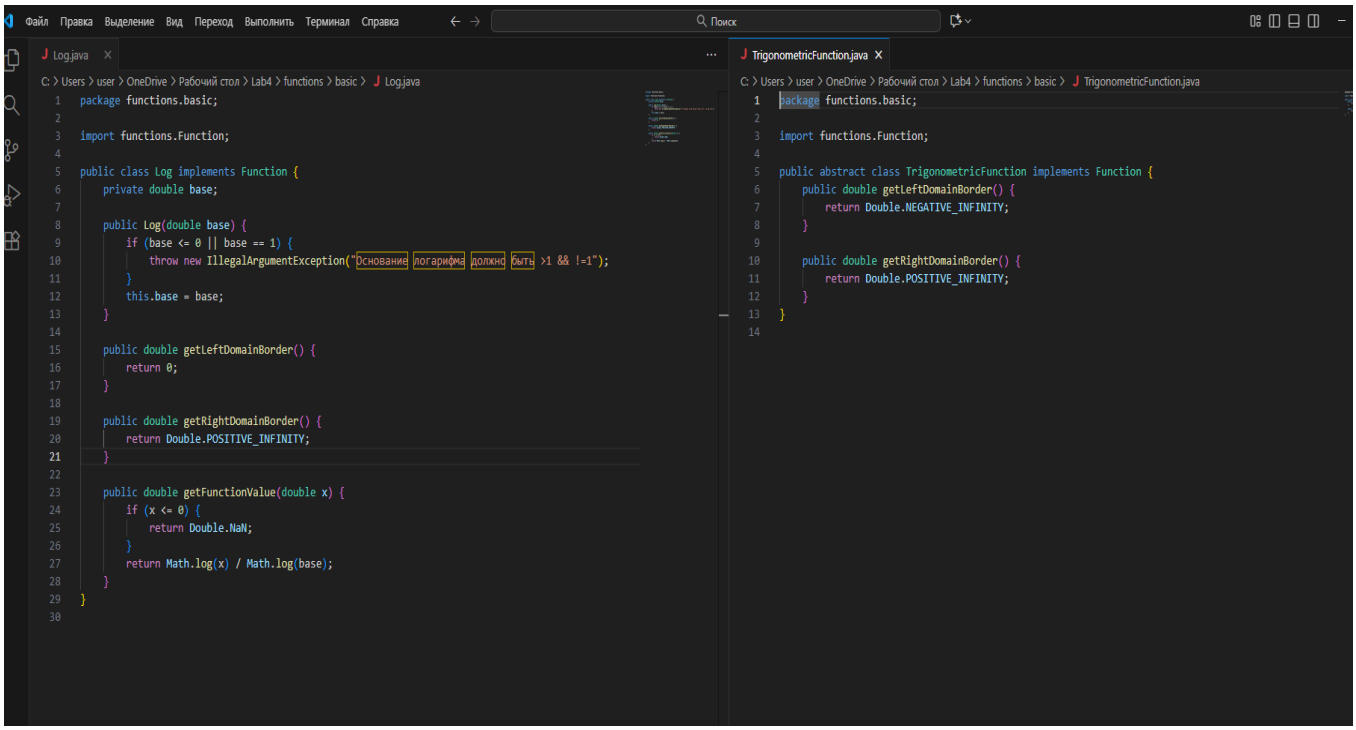


Рисунок 5,6,7-Снимок Экрана с конечным результатом для задания 3

Задание 4

Ход выполнения работы:

1)Создаем пакет functions.meta для комбинированных функций

2)Создаем класс Sum для суммы двух функций:

Конструктор принимает две функции

Область определения - пересечение областей исходных функций

3)Создаем класс Power для возведения функции в степень:

Конструктор принимает функцию и степень

Область определения совпадает с исходной функцией

4)Создаем класс Scale для масштабирования:

Конструктор принимает функцию и коэффициенты масштабирования

Масштабирует область определения и значения функции

5)Создаем класс Shift для сдвига:

Конструктор принимает функцию и величины сдвига

6)Создаем класс Composition для композиции функций:

Конструктор принимает две функции

Область определения совпадает с первой функцией

7)Создаем класс Mult для произведения двух функций:

Аналогично классу Sum

Sum.java

C: > Users > user > OneDrive > Рабочий стол > Lab4 > functions > meta > Sum.java

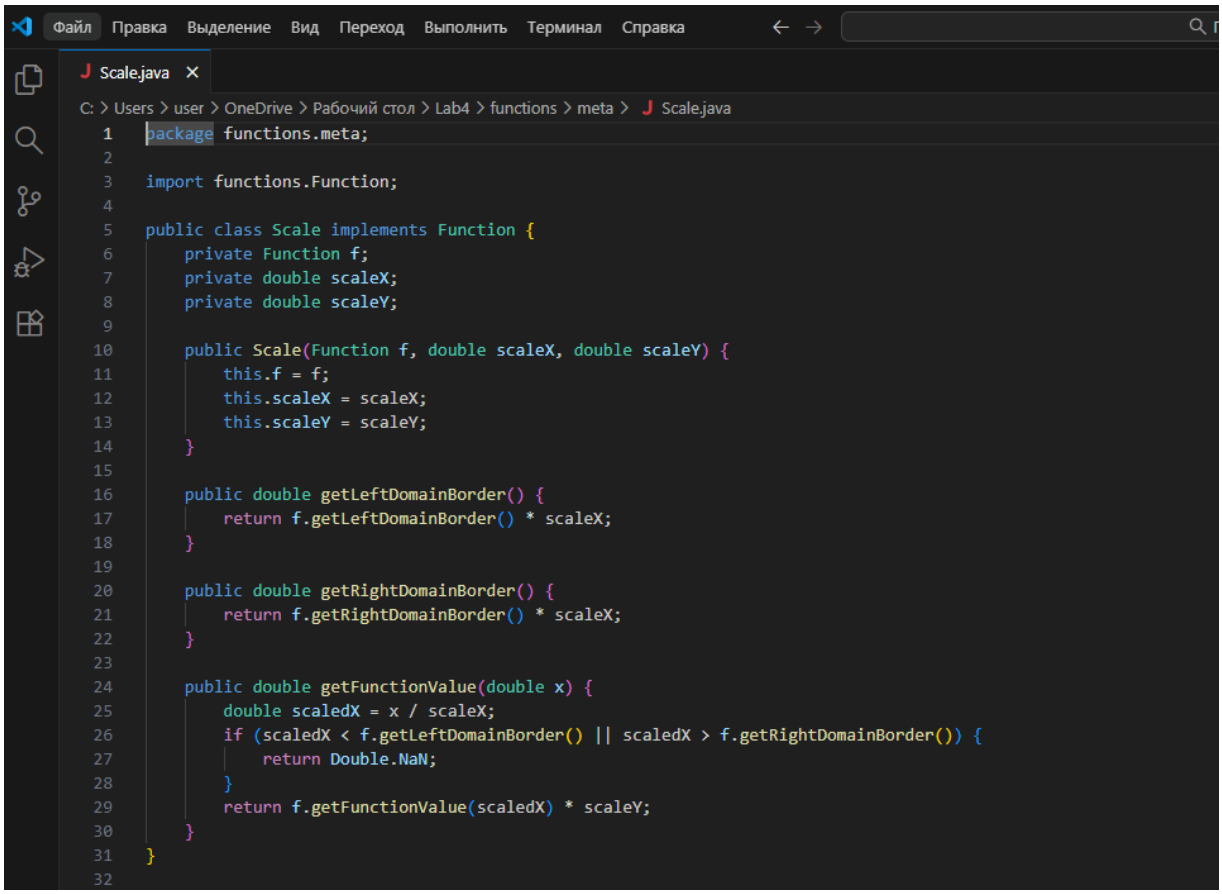
```java
package functions.meta;

import functions.Function;

public class Sum implements Function {
    private Function f1;
    private Function f2;

    public Sum(Function f1, Function f2) {
        this.f1 = f1;
        this.f2 = f2;
    }

    public double getLeftDomainBorder() {
        return Math.max(f1.getLeftDomainBorder(), f2.getLeftDomainBorder());
    }

    public double getRightDomainBorder() {
        return Math.min(f1.getRightDomainBorder(), f2.getRightDomainBorder());
    }

    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        return f1.getFunctionValue(x) + f2.getFunctionValue(x);
    }
}
```

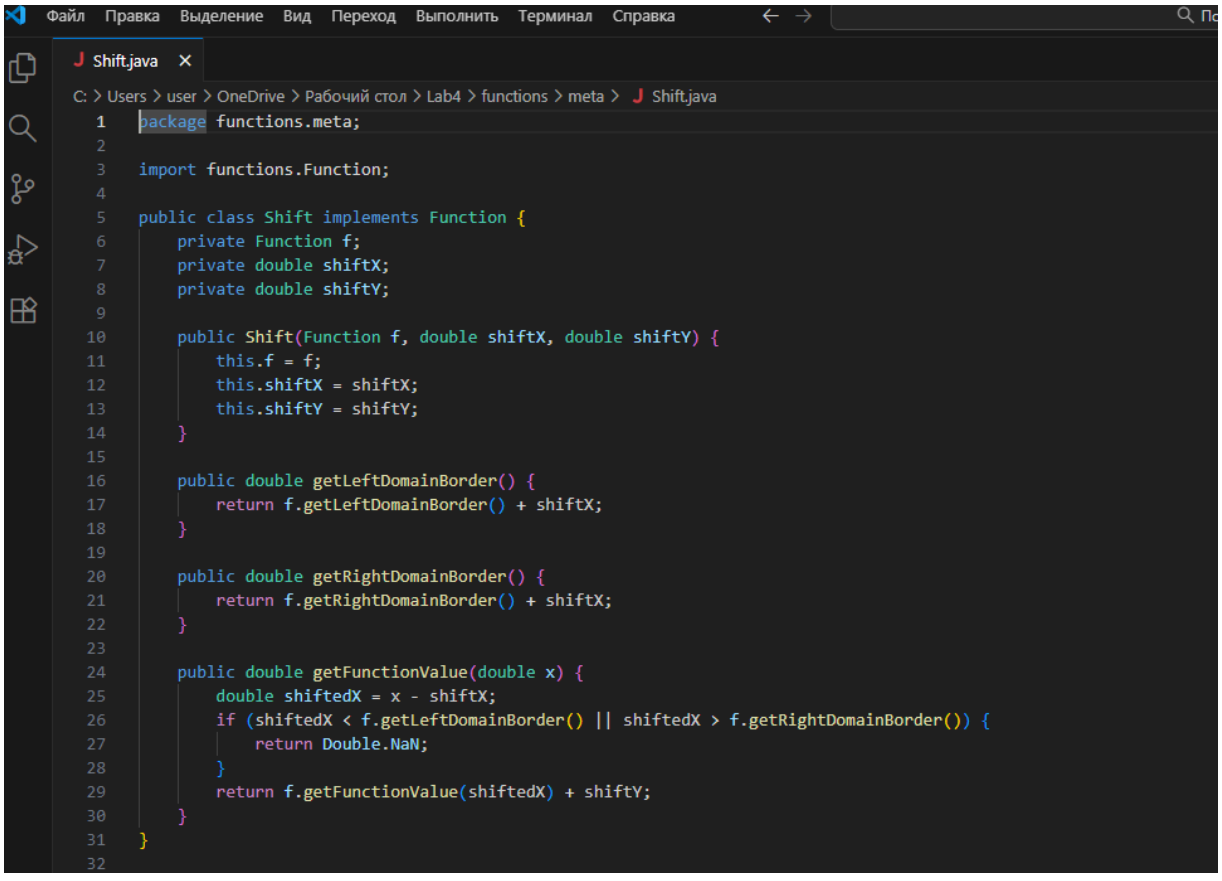
Файл Правка Выделение Вид Переход Выполнить Терминал Справка

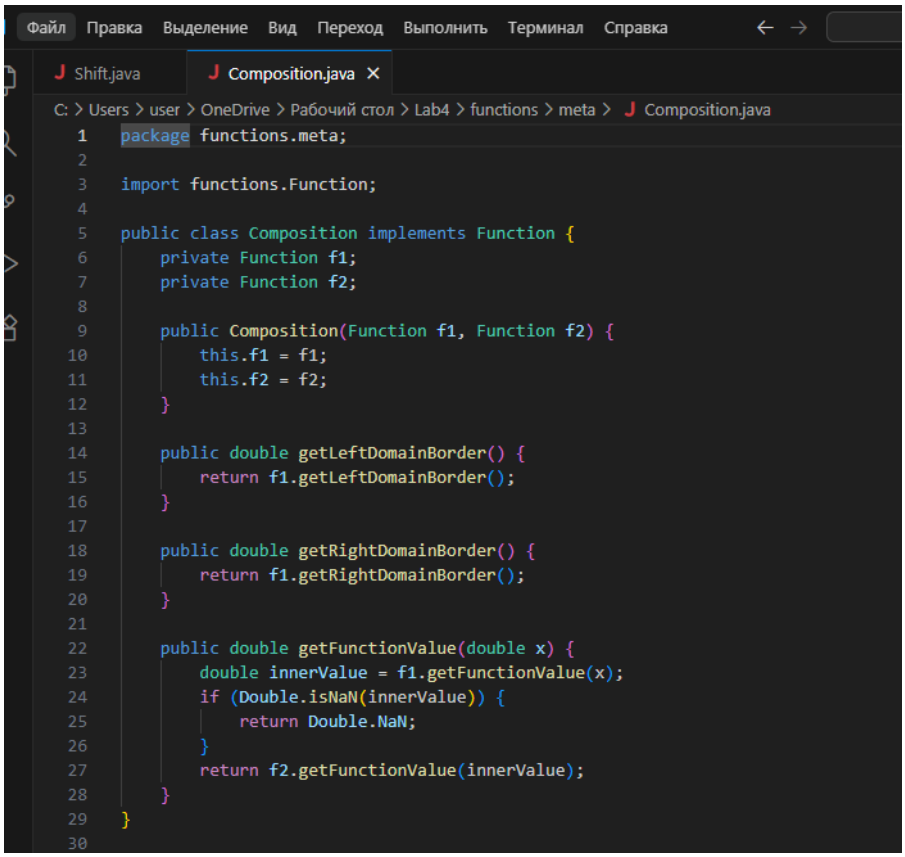
Sum.java Power.java

C: > Users > user > OneDrive > Рабочий стол > Lab4 > functions > meta > Power.java

```java
package functions.meta;//возведение одной функции в степень другой

import functions.Function;

public class Power implements Function {
    private Function f;
    private double power;

    public Power(Function f, double power) {
        this.f = f;
        this.power = power;
    }

    public double getLeftDomainBorder() {
        return f.getLeftDomainBorder();
    }

    public double getRightDomainBorder() {
        return f.getRightDomainBorder();
    }

    public double getFunctionValue(double x) {
        double value = f.getFunctionValue(x);
        if (Double.isNaN(value)) {
            return Double.NaN;
        }
        return Math.pow(value, power);
    }
}
```


Файл Правка Выделение Вид Переход Выполнить Терминал Справка
Scale.java
C: > Users > user > OneDrive > Рабочий стол > Lab4 > functions > meta > Scale.java
package functions.meta;
import functions.Function;
public class Scale implements Function {
private Function f;
private double scaleX;
private double scaleY;
public Scale(Function f, double scaleX, double scaleY) {
this.f = f;
this.scaleX = scaleX;
this.scaleY = scaleY;
}
public double getLeftDomainBorder() {
return f.getLeftDomainBorder() * scaleX;
}
public double getRightDomainBorder() {
return f.getRightDomainBorder() * scaleX;
}
public double getFunctionValue(double x) {
double scaledX = x / scaleX;
if (scaledX < f.getLeftDomainBorder() || scaledX > f.getRightDomainBorder()) {
return Double.NaN;
}
return f.getFunctionValue(scaledX) * scaleY;
}
}


Файл Правка Выделение Вид Переход Выполнить Терминал Справка
Shift.java
C: > Users > user > OneDrive > Рабочий стол > Lab4 > functions > meta > Shift.java
package functions.meta;
import functions.Function;
public class Shift implements Function {
private Function f;
private double shiftX;
private double shiftY;
public Shift(Function f, double shiftX, double shiftY) {
this.f = f;
this.shiftX = shiftX;
this.shiftY = shiftY;
}
public double getLeftDomainBorder() {
return f.getLeftDomainBorder() + shiftX;
}
public double getRightDomainBorder() {
return f.getRightDomainBorder() + shiftX;
}
public double getFunctionValue(double x) {
double shiftedX = x - shiftX;
if (shiftedX < f.getLeftDomainBorder() || shiftedX > f.getRightDomainBorder()) {
return Double.NaN;
}
return f.getFunctionValue(shiftedX) + shiftY;
}
}



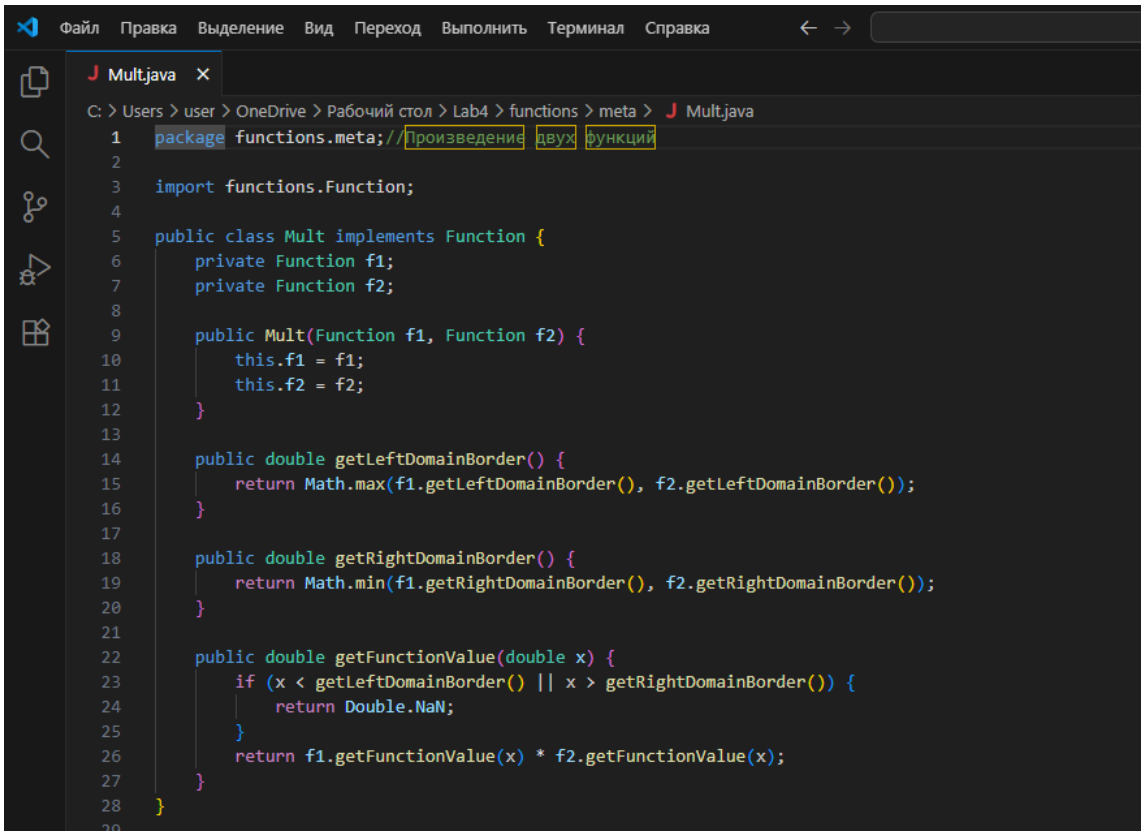


Рисунок 8-13-Снимок Экрана с конечным результатом для задания 4.

Задание 5

Ход выполнения работы:

1)Создаем класс Functions в пакете functions

Делаем конструктор приватным (утилитный класс)

Добавляем статические методы:

shift() - возвращает функцию сдвига

scale() - возвращает масштабированную функцию

power() - возвращает функцию в степени

sum() - возвращает сумму функций

mult() - возвращает произведение функций

composition() - возвращает композицию функций

2)В методах используем соответствующие классы из пакета functions.meta.

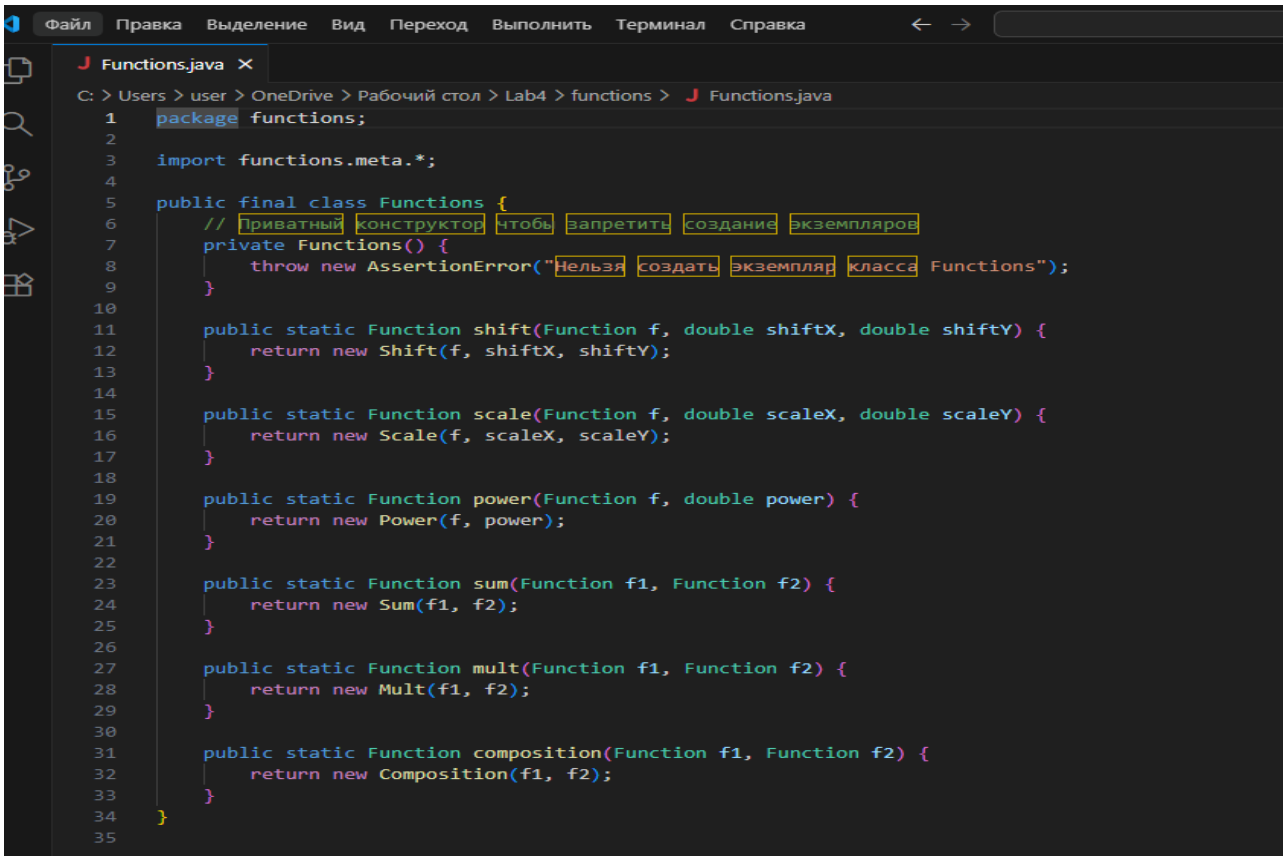


Рисунок 14-Снимок Экрана с конечным результатом для задания 5.

Задание 6

Ход выполнения работы:

1)Создаем класс TabulatedFunctions в пакете functions

Делаем конструктор приватным (утилитный класс)

2)Реализуем метод tabulate():

Проверяем, что границы табулирования входят в область определения функции. Проверяем, что количество точек не менее 2

4)Создаем массив точек с равномерным шагом. Вычисляем значения функции в этих точках. Возвращаем табулированную функцию

TabulatedFunctions.java

C: > Users > user > OneDrive > Рабочий стол > Lab4 > functions > TabulatedFunctions.java

```java
package functions;

import java.io.*;
import java.util.StringTokenizer;

public final class TabulatedFunctions {
    // Приватный конструктор чтобы запретить создание экземпляров
    private TabulatedFunctions() {
        throw new AssertionError("Нельзя создать экземпляр класса TabulatedFunctions");
    }

    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount)
        if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
            throw new IllegalArgumentException("Границы табулирования выходят за область определения функции"
        }
        if (pointsCount < 2) {
            throw new IllegalArgumentException("Количество точек должно быть не менее 2");
        }

        // Создаем массив точек
        FunctionPoint[] points = new FunctionPoint[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);

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

        // Возвращаем табулированную функцию (пока используем ArrayTabulatedFunction)
        return new ArrayTabulatedFunction(points);
    }
    public static double getFunctionValue(TabulatedFunction function, double x) {
        return function.getFunctionValue(x);
    }

    public static double getLeftDomainBorder(TabulatedFunction function) {
        return function.getLeftDomainBorder();
    }

    public static double getRightDomainBorder(TabulatedFunction function) {
        return function.getRightDomainBorder();
    }
```

Рисунок 15-Снимок Экрана с конечным результатом для задания 6.

Задание 7

Ход выполнения работы:

1) Добавляем в TabulatedFunctions метод outputTabulatedFunction():

Использует DataOutputStream для записи в байтовый поток

Записывает количество точек и координаты (x,y) каждой точки

2)Добавляем метод inputTabulatedFunction():

Использует DataInputStream для чтения из байтового потока

Читает количество точек и создает массив точек

3)Добавляем метод writeTabulatedFunction():

Использует PrintWriter для записи в символьный поток

Записывает данные в текстовом формате с пробелами-разделителями

4)Добавляем метод readTabulatedFunction():

Использует StreamTokenizer для чтения из символьного потока

Парсит текстовые данные и создает табулированную функцию

Обрабатываем IOException, оборачивая в RuntimeException

```java
    // Метод вывода в байтовый поток
    public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) {
        try (DataOutputStream dos = new DataOutputStream(out)) {
            dos.writeInt(function.getPointsCount());
            for (int i = 0; i < function.getPointsCount(); i++) {
                FunctionPoint point = function.getPoint(i);
                dos.writeDouble(point.getX());
                dos.writeDouble(point.getY());
            }
        } catch (IOException e) {
            // Пробрасываем RuntimeException, так как IOException - проверяемое исключение
            throw new RuntimeException("Ошибка при выводе функции в поток", e);
        }
    }

    // Метод ввода из байтового потока
    public static TabulatedFunction inputTabulatedFunction(InputStream in) {
        try (DataInputStream dis = new DataInputStream(in)) {
            int pointsCount = dis.readInt();
            FunctionPoint[] points = new FunctionPoint[pointsCount];

            for (int i = 0; i < pointsCount; i++) {
                double x = dis.readDouble();
                double y = dis.readDouble();
                points[i] = new FunctionPoint(x, y);
            }

            return new ArrayTabulatedFunction(points);
        } catch (IOException e) {
            throw new RuntimeException("Ошибка при чтении функции из потока", e);
        }
    }

    // Метод записи в символьный поток
    public static void writeTabulatedFunction(TabulatedFunction function, Writer out) {
        try (PrintWriter writer = new PrintWriter(out)) {
            writer.print(function.getPointsCount() + " ");
            for (int i = 0; i < function.getPointsCount(); i++) {
                FunctionPoint point = function.getPoint(i);
                writer.print(point.getX() + " " + point.getY() + " ");
            }
        }
        // PrintWriter не бросает IOException в методах print/println
    }
```

```java
    // Метод чтения из символьного потока
    public static TabulatedFunction readTabulatedFunction(Reader in) {
        try {
            StreamTokenizer tokenizer = new StreamTokenizer(in);
            tokenizer.parseNumbers();

            // Читаем количество точек
            if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
                throw new RuntimeException("Ожидалось количество точек");
            }
            int pointsCount = (int) tokenizer.nval;

            FunctionPoint[] points = new FunctionPoint[pointsCount];

            for (int i = 0; i < pointsCount; i++) {
                if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
                    throw new RuntimeException("Ожидалась координата x");
                }
                double x = tokenizer.nval;

                if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
                    throw new RuntimeException("Ожидалась координата y");
                }
                double y = tokenizer.nval;

                points[i] = new FunctionPoint(x, y);
            }

            return new ArrayTabulatedFunction(points);
        } catch (IOException e) {
            throw new RuntimeException("Ошибка при чтении функции из потока", e);
        }
    }
}
```

Рисунок 16,17-Снимок экрана с конечным результатом для задания 7.

Задание 8

Ход выполнения работы:

1)Создаем объекты Sin и Cos, выводим их значения на отрезке [0, π]

2)Создаем табулированные аналоги с помощью TabulatedFunctions.tabulate()

3)Сравниваем значения исходных и табулированных функций

4)Создаем функцию суммы квадратов табулированных синуса и косинуса

5)Исследуем влияние количества точек на точность

6)Тестируем экспоненту:

Создаем табулированную экспоненту

Записываем в текстовый файл с помощью writeTabulatedFunction()

Читаем из файла с помощью readTabulatedFunction()

Сравниваем исходную и считанную функции

7)Тестируем логарифм:

Создаем табулированный логарифм

Записываем в бинарный файл с помощью outputTabulatedFunction()

Читаем из файла с помощью inputTabulatedFunction()

Сравниваем исходную и считанную функции

8)Анализируем файлы разных форматов, сравниваем их преимущества и недостатки

```
=== ТЕСТИРОВАНИЕ ВСЕХ ФУНКЦИЙ ===

1. ТЕСТИРОВАНИЕ SIN И COS
Sin на отрезке [0, 3.14]:
sin(0,0)=0,000000       sin(0,1)=0,099833       sin(0,2)=0,198669       sin(0,3)=0,295520       sin(0,4)=0,389418
sin(0,5)=0,479426       sin(0,6)=0,564642       sin(0,7)=0,644218       sin(0,8)=0,717356       sin(0,9)=0,783327
sin(1,0)=0,841471       sin(1,1)=0,891207       sin(1,2)=0,932039       sin(1,3)=0,963558       sin(1,4)=0,985450
sin(1,5)=0,997495       sin(1,6)=0,999574       sin(1,7)=0,991665       sin(1,8)=0,973848       sin(1,9)=0,946300
sin(2,0)=0,909297       sin(2,1)=0,863209       sin(2,2)=0,808496       sin(2,3)=0,745705       sin(2,4)=0,675463
sin(2,5)=0,598472       sin(2,6)=0,515501       sin(2,7)=0,427380       sin(2,8)=0,334988       sin(2,9)=0,239249
sin(3,0)=0,141120       sin(3,1)=0,041581

Cos на отрезке [0, 3.14]:
cos(0,0)=1,000000       cos(0,1)=0,995004       cos(0,2)=0,980067       cos(0,3)=0,955336       cos(0,4)=0,921061
cos(0,5)=0,877583       cos(0,6)=0,825336       cos(0,7)=0,764842       cos(0,8)=0,696707       cos(0,9)=0,621610
cos(1,0)=0,540302       cos(1,1)=0,453596       cos(1,2)=0,362358       cos(1,3)=0,267499       cos(1,4)=0,169967
cos(1,5)=0,070737       cos(1,6)=-0,029200      cos(1,7)=-0,128844      cos(1,8)=-0,227202      cos(1,9)=-0,323290
cos(2,0)=-0,416147      cos(2,1)=-0,504846      cos(2,2)=-0,588501      cos(2,3)=-0,666276      cos(2,4)=-0,737394
cos(2,5)=-0,801144      cos(2,6)=-0,856889      cos(2,7)=-0,904072      cos(2,8)=-0,942222      cos(2,9)=-0,970958
cos(3,0)=-0,989992      cos(3,1)=-0,999135


2. ТАБУЛИРОВАННЫЕ АНАЛОГИ
Сравнение исходного и табулированного Sin:
x=0,0: Orig=0,000000 Tabulated=0,000000 differense=0,000000    x=0,1: Orig=0,099833 Tabulated=0,097982 differense=0,001852    x=0,2: Orig=0,198669 Tabulated=0,195963 differense=0,002706
x=0,3: Orig=0,295520 Tabulated=0,293945 differense=0,001576    x=0,4: Orig=0,389418 Tabulated=0,385907 differense=0,003512    x=0,5: Orig=0,479426 Tabulated=0,472070 differense=0,007355
x=0,6: Orig=0,564642 Tabulated=0,558234 differense=0,006409    x=0,7: Orig=0,644218 Tabulated=0,643982 differense=0,000235    x=0,8: Orig=0,717356 Tabulated=0,707935 differense=0,009421
x=0,9: Orig=0,783327 Tabulated=0,771888 differense=0,011439    x=1,0: Orig=0,841471 Tabulated=0,835841 differense=0,005630    x=1,1: Orig=0,891207 Tabulated=0,883993 differense=0,007214
x=1,2: Orig=0,932039 Tabulated=0,918022 differense=0,014017    x=1,3: Orig=0,963558 Tabulated=0,952051 differense=0,011508    x=1,4: Orig=0,985450 Tabulated=0,984808 differense=0,000642
x=1,5: Orig=0,997495 Tabulated=0,984808 differense=0,012687    x=1,6: Orig=0,999574 Tabulated=0,984808 differense=0,014766    x=1,7: Orig=0,991665 Tabulated=0,984808 differense=0,006857
x=1,8: Orig=0,973848 Tabulated=0,966204 differense=0,007644    x=1,9: Orig=0,946300 Tabulated=0,932175 differense=0,014125    x=2,0: Orig=0,909297 Tabulated=0,898147 differense=0,011151
x=2,1: Orig=0,863209 Tabulated=0,862441 differense=0,000768    x=2,2: Orig=0,808496 Tabulated=0,798488 differense=0,010008    x=2,3: Orig=0,745705 Tabulated=0,734535 differense=0,011170
x=2,4: Orig=0,675463 Tabulated=0,670582 differense=0,004881    x=2,5: Orig=0,598472 Tabulated=0,594072 differense=0,004401    x=2,6: Orig=0,515501 Tabulated=0,507908 differense=0,007593
x=2,7: Orig=0,427380 Tabulated=0,421745 differense=0,005635    x=2,8: Orig=0,334988 Tabulated=0,334698 differense=0,000290    x=2,9: Orig=0,239249 Tabulated=0,236716 differense=0,002533
x=3,0: Orig=0,141120 Tabulated=0,138735 differense=0,002385    x=3,1: Orig=0,041581 Tabulated=0,040753 differense=0,000828
Сравнение исходного и табулированного Cos:
x=0,0: Orig=1,000000 Tabulated=0,000000 differense=1,000000    x=0,1: Orig=0,995004 Tabulated=0,097982 differense=0,897023    x=0,2: Orig=0,980067 Tabulated=0,195963 differense=0,784103
x=0,3: Orig=0,955336 Tabulated=0,293945 differense=0,661392    x=0,4: Orig=0,921061 Tabulated=0,385907 differense=0,535154    x=0,5: Orig=0,877583 Tabulated=0,472070 differense=0,405512
x=0,6: Orig=0,825336 Tabulated=0,558234 differense=0,267102    x=0,7: Orig=0,764842 Tabulated=0,643982 differense=0,120860    x=0,8: Orig=0,696707 Tabulated=0,707935 differense=-0,011229
x=0,9: Orig=0,621610 Tabulated=0,771888 differense=-0,150278   x=1,0: Orig=0,540302 Tabulated=0,835841 differense=-0,295539   x=1,1: Orig=0,453596 Tabulated=0,883993 differense=-0,430397
x=1,2: Orig=0,362358 Tabulated=0,918022 differense=-0,555664   x=1,3: Orig=0,267499 Tabulated=0,952051 differense=-0,684552   x=1,4: Orig=0,169967 Tabulated=0,984808 differense=-0,814841
x=1,5: Orig=0,070737 Tabulated=0,984808 differense=-0,914071   x=1,6: Orig=-0,029200 Tabulated=0,984808 differense=-1,014007  x=1,7: Orig=-0,128844 Tabulated=0,984808 differense=-1,113652
x=1,8: Orig=-0,227202 Tabulated=0,966204 differense=-1,193406  x=1,9: Orig=-0,323290 Tabulated=0,932175 differense=-1,255465  x=2,0: Orig=-0,416147 Tabulated=0,898147 differense=-1,314294
x=2,1: Orig=-0,504846 Tabulated=0,862441 differense=-1,367287  x=2,2: Orig=-0,588501 Tabulated=0,798488 differense=-1,386989  x=2,3: Orig=-0,666276 Tabulated=0,734535 differense=-1,400811
x=2,4: Orig=-0,737394 Tabulated=0,670582 differense=-1,407976  x=2,5: Orig=-0,801144 Tabulated=0,594072 differense=-1,395215  x=2,6: Orig=-0,856889 Tabulated=0,507908 differense=-1,364797
x=2,7: Orig=-0,904072 Tabulated=0,421745 differense=-1,325817  x=2,8: Orig=-0,942222 Tabulated=0,334698 differense=-1,276920  x=2,9: Orig=-0,970958 Tabulated=0,236716 differense=-1,207674
x=3,0: Orig=-0,989992 Tabulated=0,138735 differense=-1,128727  x=3,1: Orig=-0,999135 Tabulated=0,040753 differense=-1,039888
```

```
3. СУММА КВАДРАТОВ SIN И COS
(sin(x))**2 + (cos(x))**2 на отрезке [0, 3.14]:
sum(0,0)=1,000000       sum(0,1)=0,975345       sum(0,2)=0,970488       sum(0,3)=0,985429       sum(0,4)=0,984968
sum(0,5)=0,970398       sum(0,6)=0,975624       sum(0,7)=0,999358       sum(0,8)=0,975073       sum(0,9)=0,970586
sum(1,0)=0,985897       sum(1,1)=0,984515       sum(1,2)=0,970314       sum(1,3)=0,975910       sum(1,4)=0,998723
sum(1,5)=0,974808       sum(1,6)=0,970691       sum(1,7)=0,986371       sum(1,8)=0,984068       sum(1,9)=0,970237
sum(2,0)=0,976203       sum(2,1)=0,998094       sum(2,2)=0,974549       sum(2,3)=0,970802       sum(2,4)=0,986852
sum(2,5)=0,983628       sum(2,6)=0,970167       sum(2,7)=0,976503       sum(2,8)=0,997473       sum(2,9)=0,974298
sum(3,0)=0,970920       sum(3,1)=0,987341


4. ВЛИЯНИЕ КОЛИЧЕСТВА ТОЧЕК НА ТОЧНОСТЬ
Точек: 5, максимальная ошибка: 0,146396
Точек: 10, максимальная ошибка: 0,029833
Точек: 20, максимальная ошибка: 0,006817
Точек: 50, максимальная ошибка: 0,001026


5. ТЕСТИРОВАНИЕ С ЭКСПОНЕНТОЙ (текстовый файл)
Экспонента записана в exp_function.txt
Экспонента прочитана из exp_function.txt
Сравнение исходной и считанной экспоненты:
x=0,0: Orig=1,000000 C=1,000000 Совпадают
x=1,0: Orig=2,718282 C=2,718282 Совпадают
x=2,0: Orig=7,389056 C=7,389056 Совпадают
x=3,0: Orig=20,085537 C=20,085537 Совпадают
x=4,0: Orig=54,598150 C=54,598150 Совпадают
x=5,0: Orig=148,413159 C=148,413159 Совпадают
x=6,0: Orig=403,428793 C=403,428793 Совпадают
x=7,0: Orig=1096,633158 C=1096,633158 Совпадают
x=8,0: Orig=2980,957987 C=2980,957987 Совпадают
x=9,0: Orig=8103,083928 C=8103,083928 Совпадают
x=10,0: Orig=22026,465795 C=22026,465795 Совпадают


6. ТЕСТИРОВАНИЕ С ЛОГАРИФМОМ (бинарный файл)
Логарифм записан в log_function.bin
Логарифм прочитан из log_function.bin
Сравнение исходной и считанной логарифма:
x=0,1: Orig=-2,302585 C=-2,302585 Совпадают
x=1,1: Orig=0,092705 C=0,092705 Совпадают
x=2,1: Orig=0,740233 C=0,740233 Совпадают
x=3,1: Orig=1,130147 C=1,130147 Совпадают
x=4,1: Orig=1,409999 C=1,409999 Совпадают
x=5,1: Orig=1,628429 C=1,628429 Совпадают
x=6,1: Orig=1,807603 C=1,807603 Совпадают
x=7,1: Orig=1,959502 C=1,959502 Совпадают
x=8,1: Orig=2,091344 C=2,091344 Совпадают
x=9,1: Orig=2,207812 C=2,207812 Совпадают
```

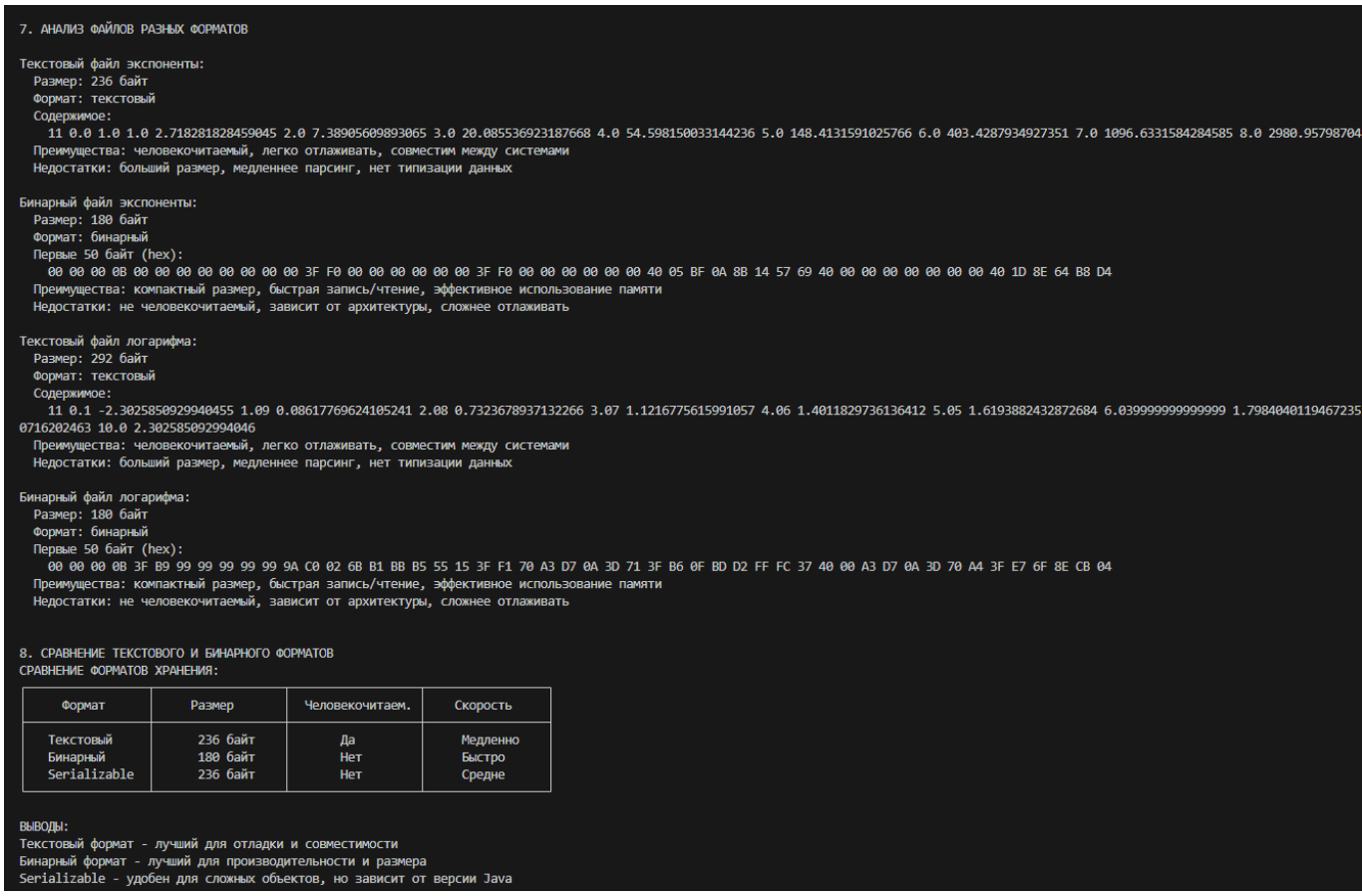


Рисунок 18-20-Снимок экрана с конечным результатом для задания 8.

Задание 9

Ход выполнения работы:

1)Делаем классы ArrayTabulatedFunction и LinkedListTabulatedFunction сериализуемыми

2)Добавляем implements Serializable и serialVersionUID

В ArrayTabulatedFunction реализуем интерфейс Externalizable:

writeExternal() записывает количество точек и координаты

readExternal() читает данные и восстанавливает объект

3)Тестируем сериализацию:

4)Создаем композицию логарифма и экспоненты

5)Табулируем композицию

6)Сериализуем в файл с помощью ObjectOutputStream

7)Десериализуем из файла с помощью ObjectInputStream

8)Сравниваем исходную и десериализованную функции

9)Анализируем файл сериализации, сравниваем с другими форматами

```
9. ТЕСТИРОВАНИЕ СЕРИАЛИЗАЦИИ
Функция сериализована в serializable_function.ser
Функция десериализована из serializable_function.ser
Сравнение исходной и десериализованной функции:
x=0,1: Orig=0,100000 D=0,100000 Совпадает       x=1,1: Orig=1,100000 D=1,100000 Совпадает       x=2,1: Orig=2,100000 D=2,100000 Совпадает
x=5,1: Orig=5,100000 D=5,100000 Совпадает       x=6,1: Orig=6,100000 D=6,100000 Совпадает       x=7,1: Orig=7,100000 D=7,100000 Совпадает

Файл сериализации (Serializable):
  Размер: 236 байт
  Формат: сериализованный объект
  Размер файла: 236 байт
  Преимущества: сохраняет структуру объектов, удобен для сложных объектов, автоматическая сериализация
  Недостатки: не человекочитаемый, зависит от версии Java, больший размер чем бинарный
PS C:\Users\user\OneDrive\Рабочий стол\Lab4>
```

Рисунок 21-Снимок экрана с конечным результатом для задания 9.