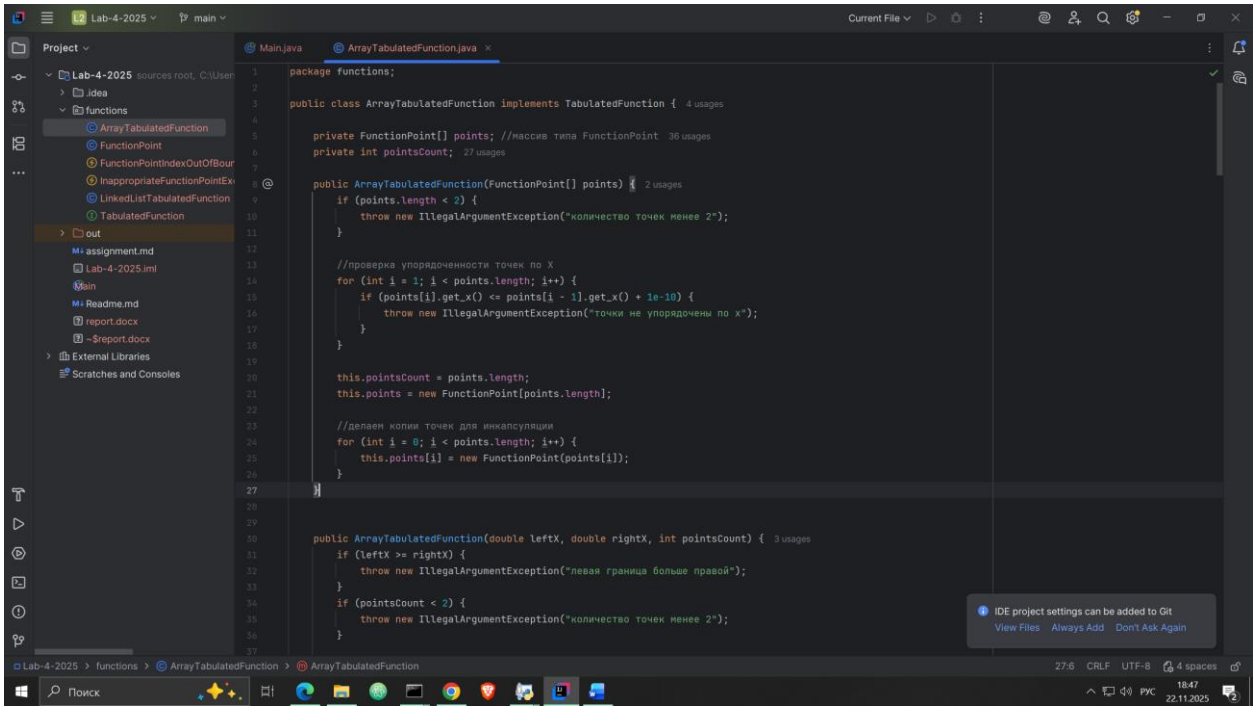
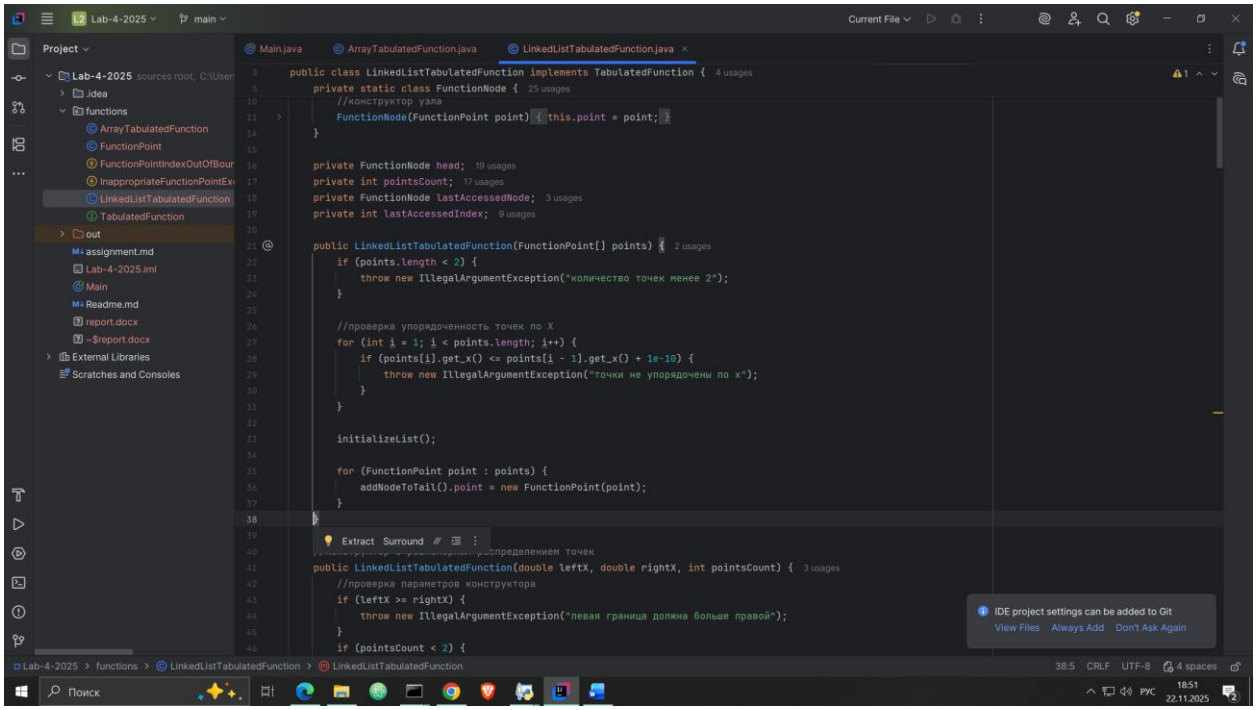
Отчёт по лабораторной работе №4

Тенигин Валерий 6204-010302D

Задание 1 – Добавлены констукторы согласно заданию



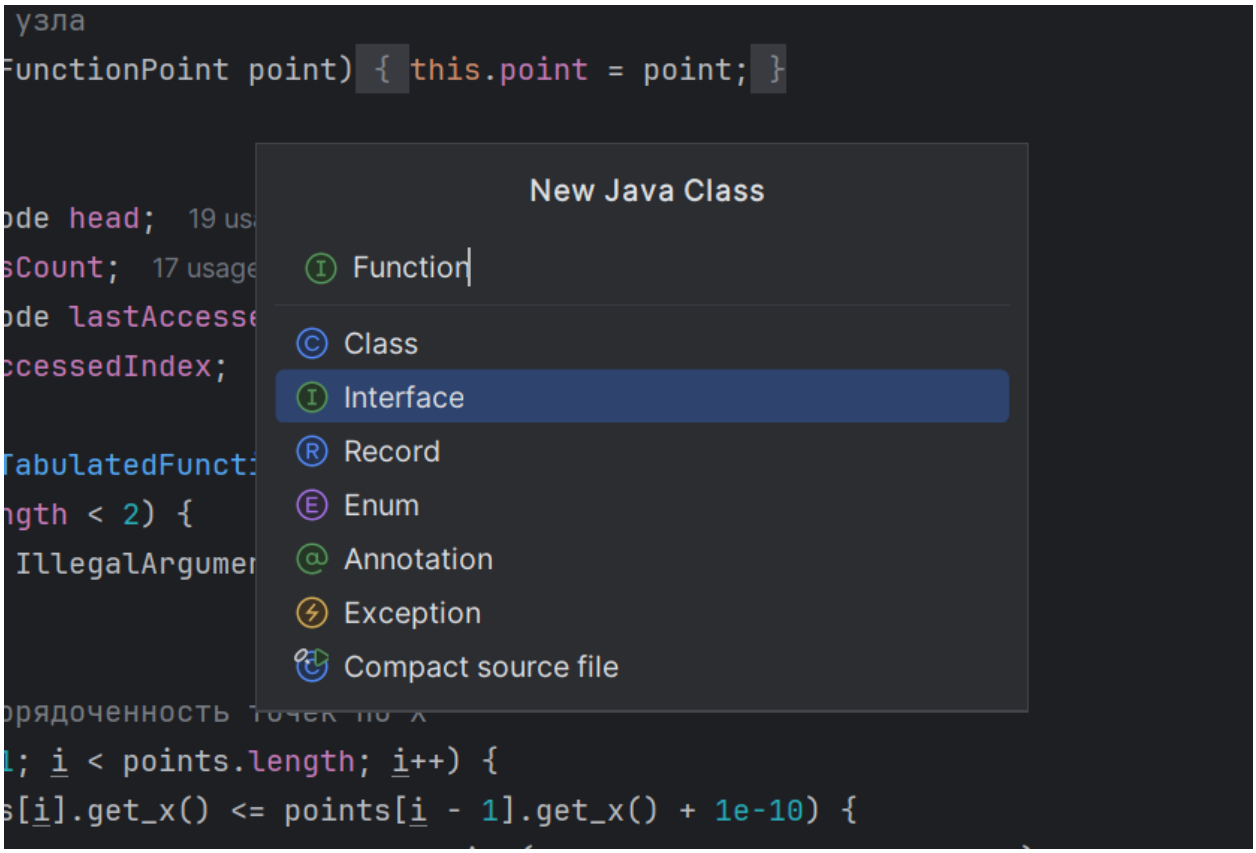


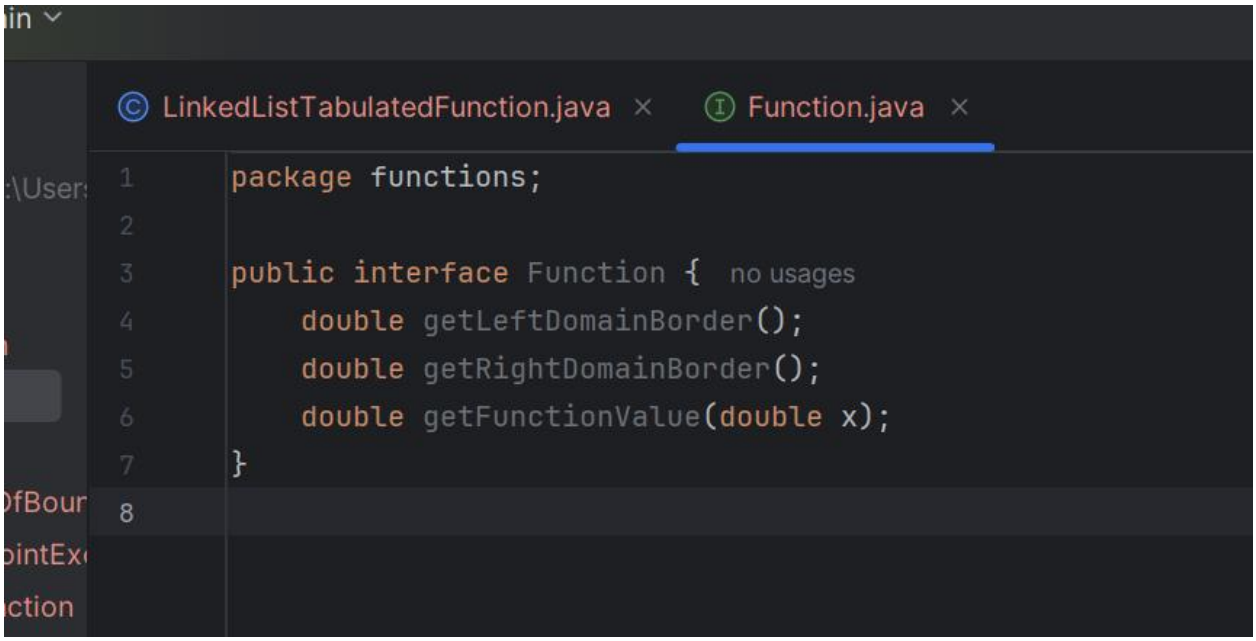
Задание 2

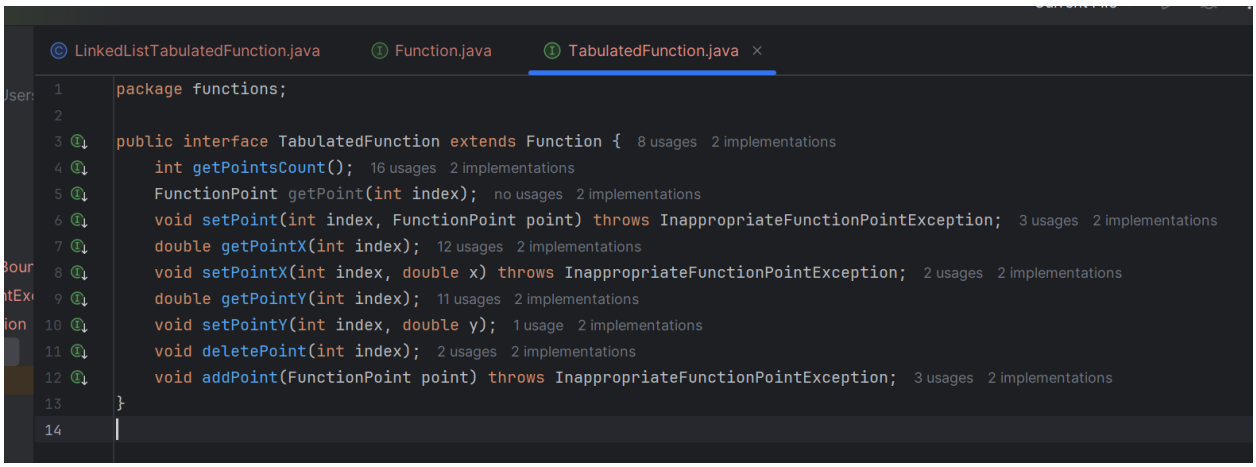
Создан интерфейс Function с методами: getLeftDomainBorder(), getRightDomainBorder(), getFunctionValue(double x)

Интерфейс TabulatedFunction теперь расширяет Function

Удалены дублирующие методы из TabulatedFunction



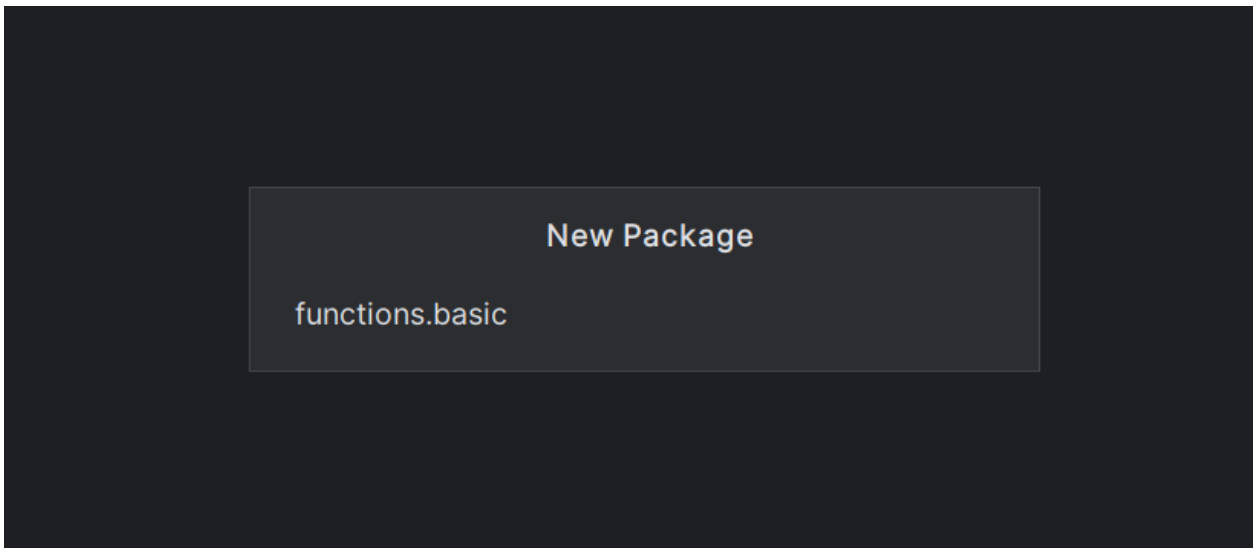


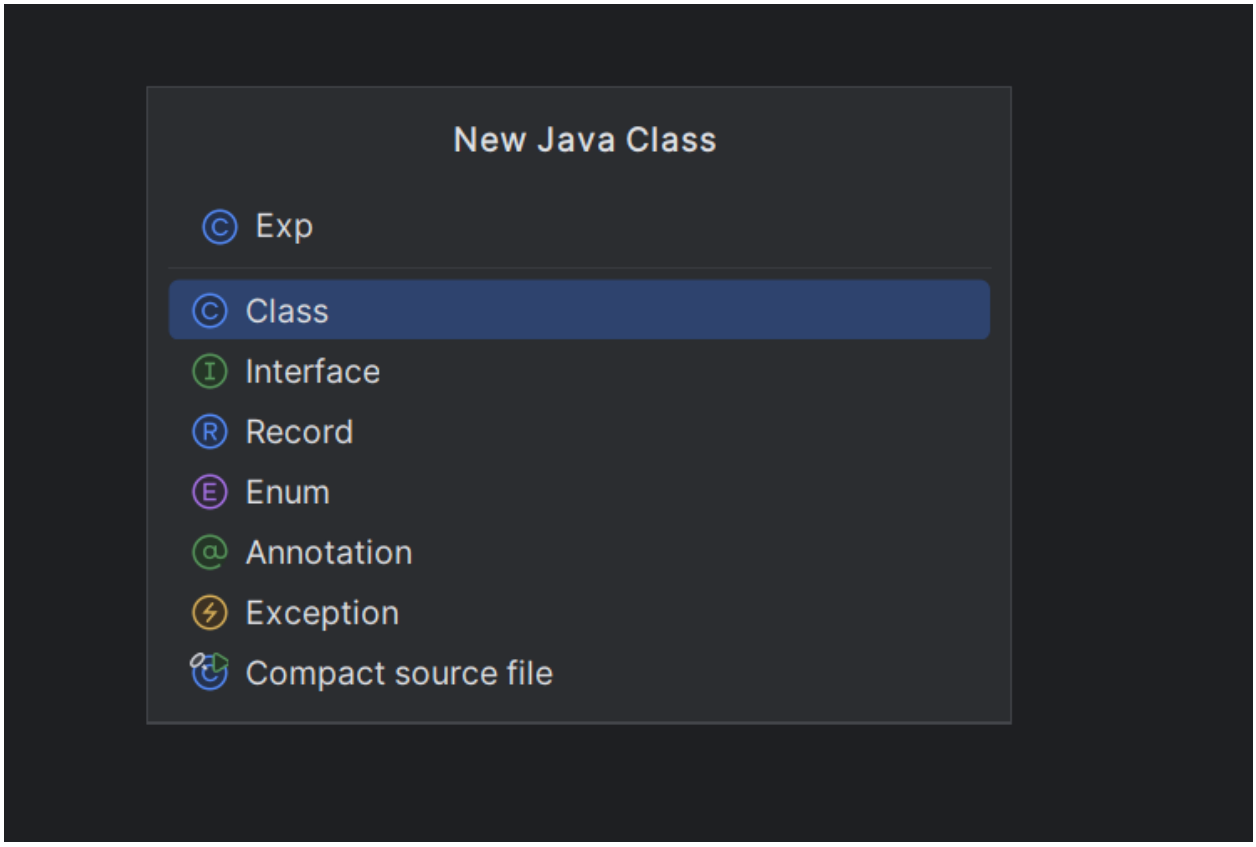


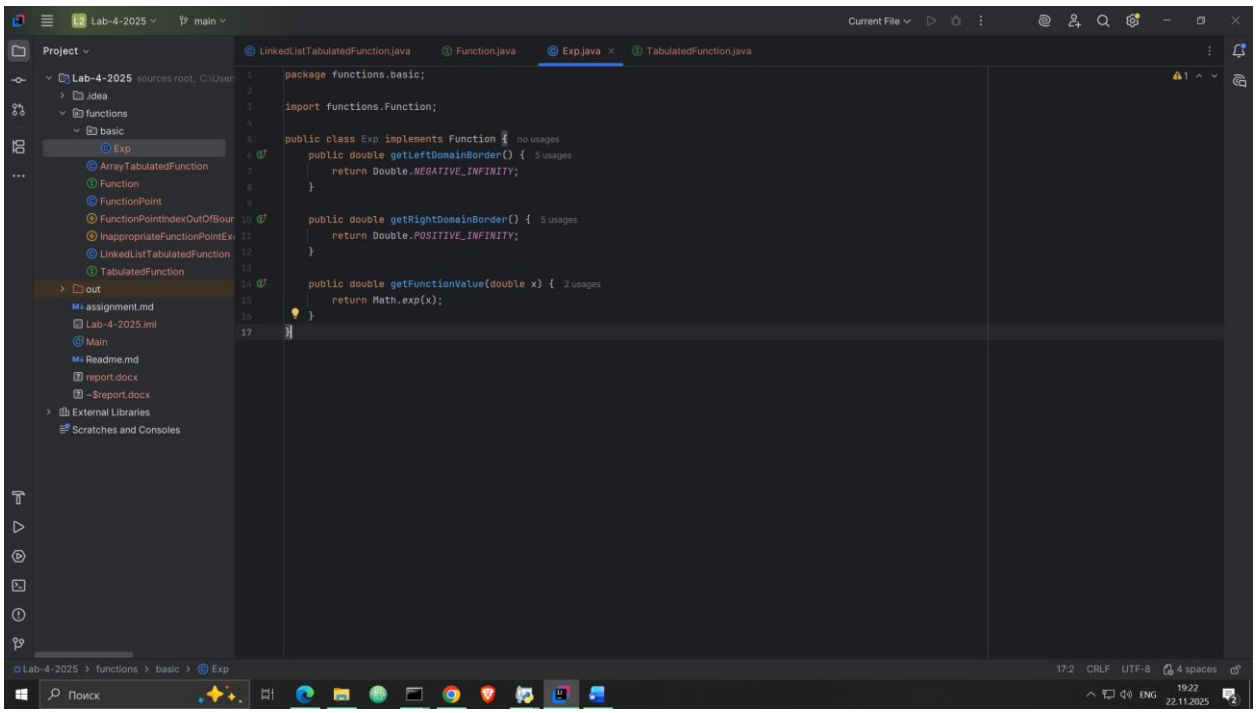
Задание 3

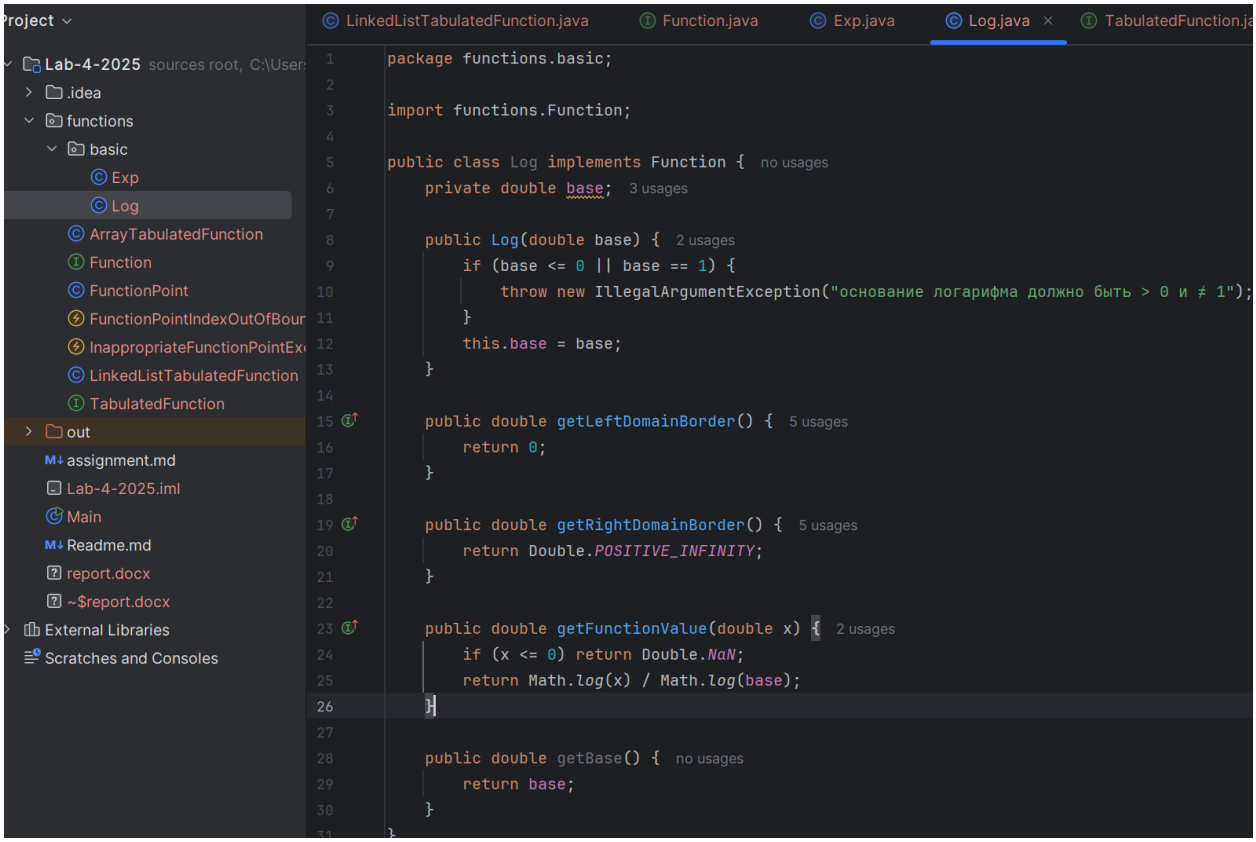
Создан пакет functions.basic

Реализованы классы – Exp(экспонента), Log (логарифм с заданным основанием), TrigonometricFunction (базовый класс для тригонометрических функций), Sin, Cos, Tan (синус, косинус, тангенс)

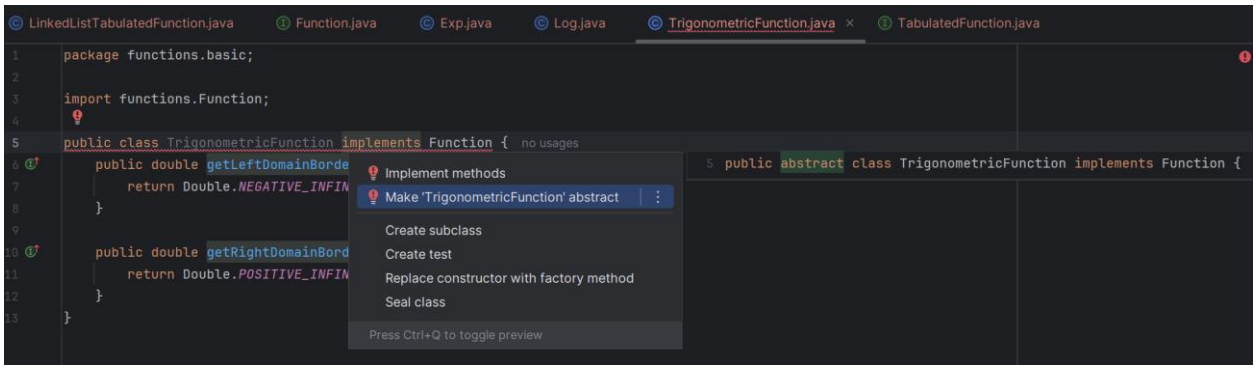

New Package
functions.basic


New Java Class
Exp
Class
Interface
Record
Enum
Annotation
Exception
Compact source file

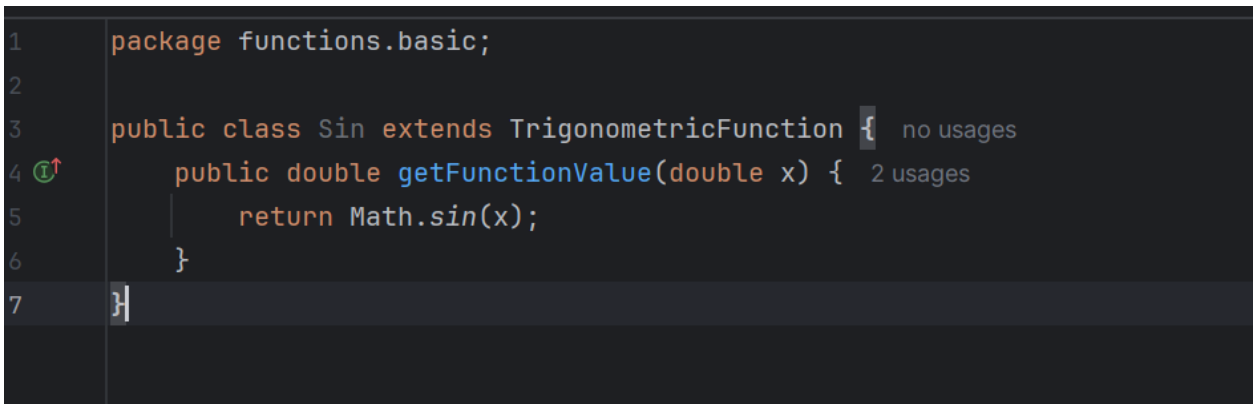

Lab-4-2025
main
Current File
Project
LinkedListTabulatedFunction.java
Function.java
Exp.java
TabulatedFunction.java
Lab-4-2025 sources root, C:\User
.idea
functions
basic
Exp
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
package functions.basic;
import functions.Function;
public class Exp implements Function { no usages
public double getLeftDomainBorder() { 5 usages
return Double.NEGATIVE_INFINITY;
}
public double getRightDomainBorder() { 5 usages
return Double.POSITIVE_INFINITY;
}
public double getFunctionValue(double x) { 2 usages
return Math.exp(x);
}
}
Lab-4-2025 › functions › basic › Exp
17:2
CRLF
UTF-8
4 spaces
Поиск
ENG
19:22
22.11.2025


Project
LinkedListTabulatedFunction.java
Function.java
Exp.java
Log.java
TabulatedFunction.ja
Lab-4-2025 sources root, C:\User
.idea
functions
basic
Exp
Log
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
package functions.basic;
import functions.Function;
public class Log implements Function { no usages
private double base; 3 usages
public Log(double base) { 2 usages
if (base <= 0 || base == 1) {
throw new IllegalArgumentException("основание логарифма должно быть > 0 и ≠ 1");
}
this.base = base;
}
public double getLeftDomainBorder() { 5 usages
return 0;
}
public double getRightDomainBorder() { 5 usages
return Double.POSITIVE_INFINITY;
}
public double getFunctionValue(double x) { 2 usages
if (x <= 0) return Double.NaN;
return Math.log(x) / Math.log(base);
}
public double getBase() { no usages
return base;
}
}

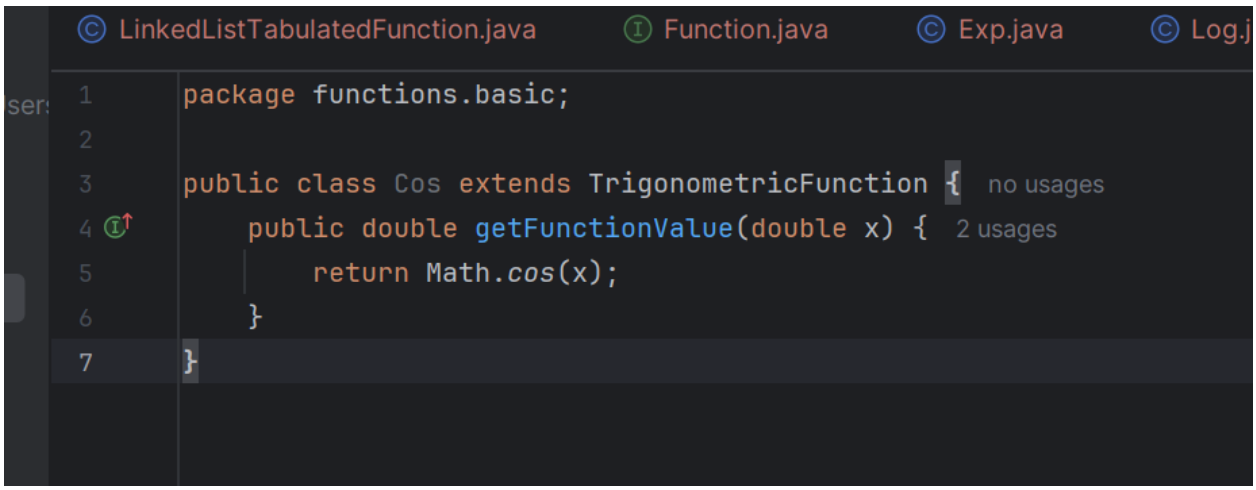
При создании TrigonometricFunction появилась ошибка, idea предложила исправить ошибку используя абстрактный класс




```
package functions.basic;

import functions.Function;

public abstract class TrigonometricFunction implements Function {  no usages
    public double getLeftDomainBorder() {  5 usages
        return Double.NEGATIVE_INFINITY;
    }

    public double getRightDomainBorder() {  5 usages
        return Double.POSITIVE_INFINITY;
    }
}
```

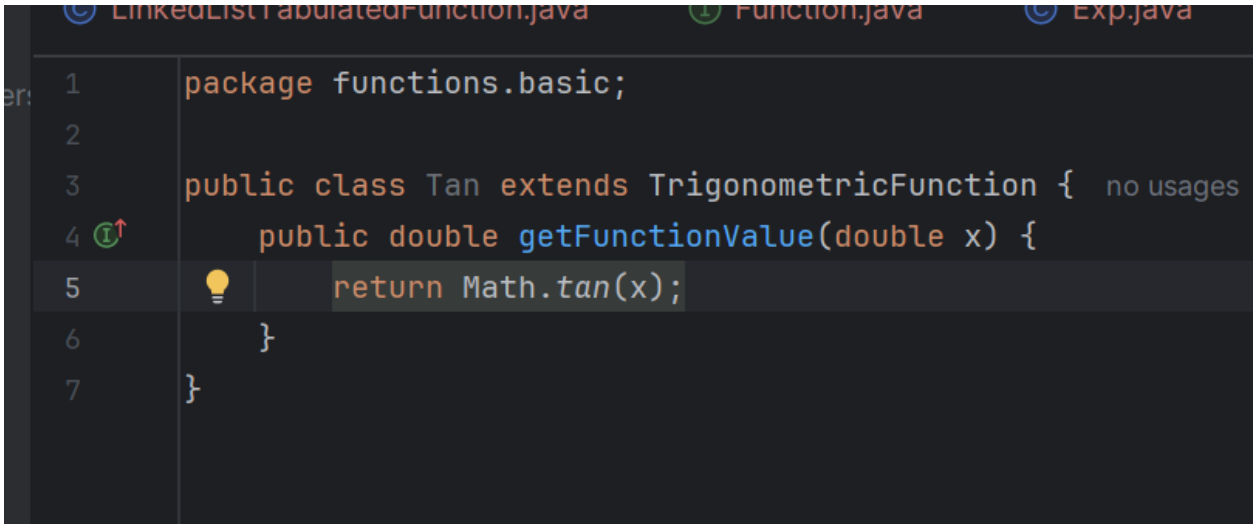


```java
package functions.basic;

public class Sin extends TrigonometricFunction {  no usages
    public double getFunctionValue(double x) {  2 usages
        return Math.sin(x);
    }
}
```



LinkedListTabulatedFunction.java Function.java Exp.java Log.j

```java
package functions.basic;

public class Cos extends TrigonometricFunction {  no usages
    public double getFunctionValue(double x) {  2 usages
        return Math.cos(x);
    }
}
```



```java
package functions.basic;

public class Tan extends TrigonometricFunction {  no usages
    public double getFunctionValue(double x) {
        return Math.tan(x);
    }
}
```

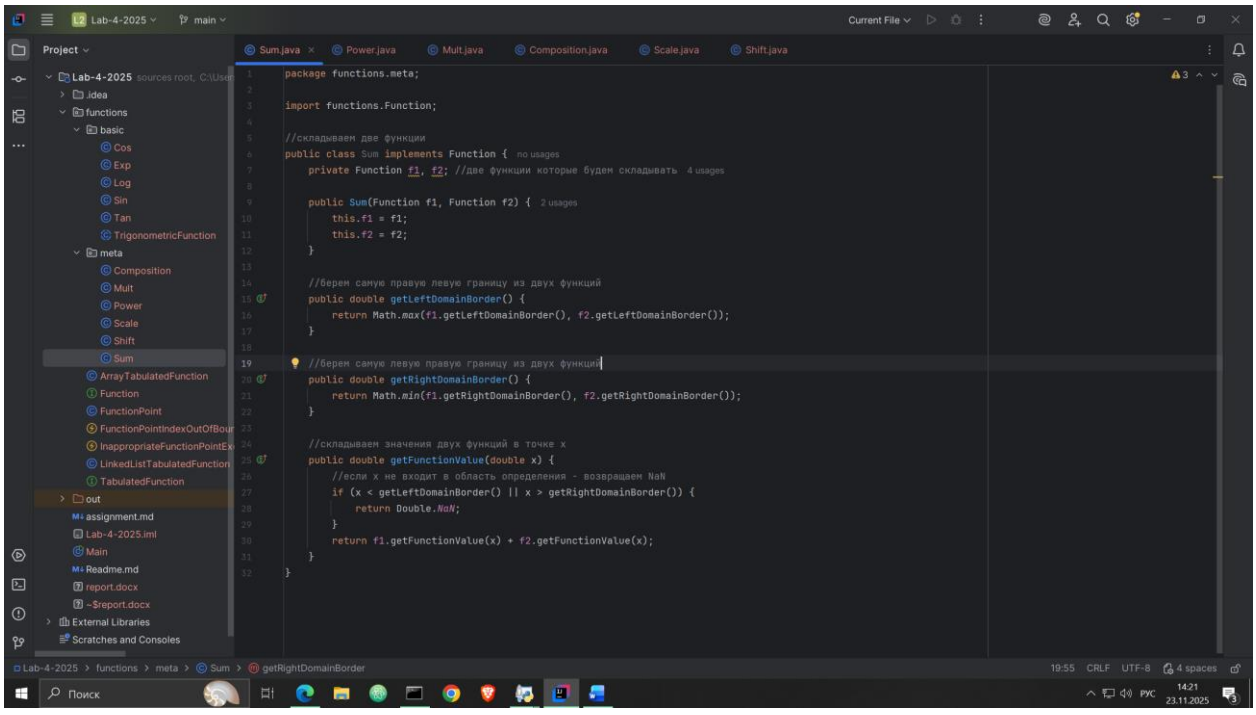
Задание 4

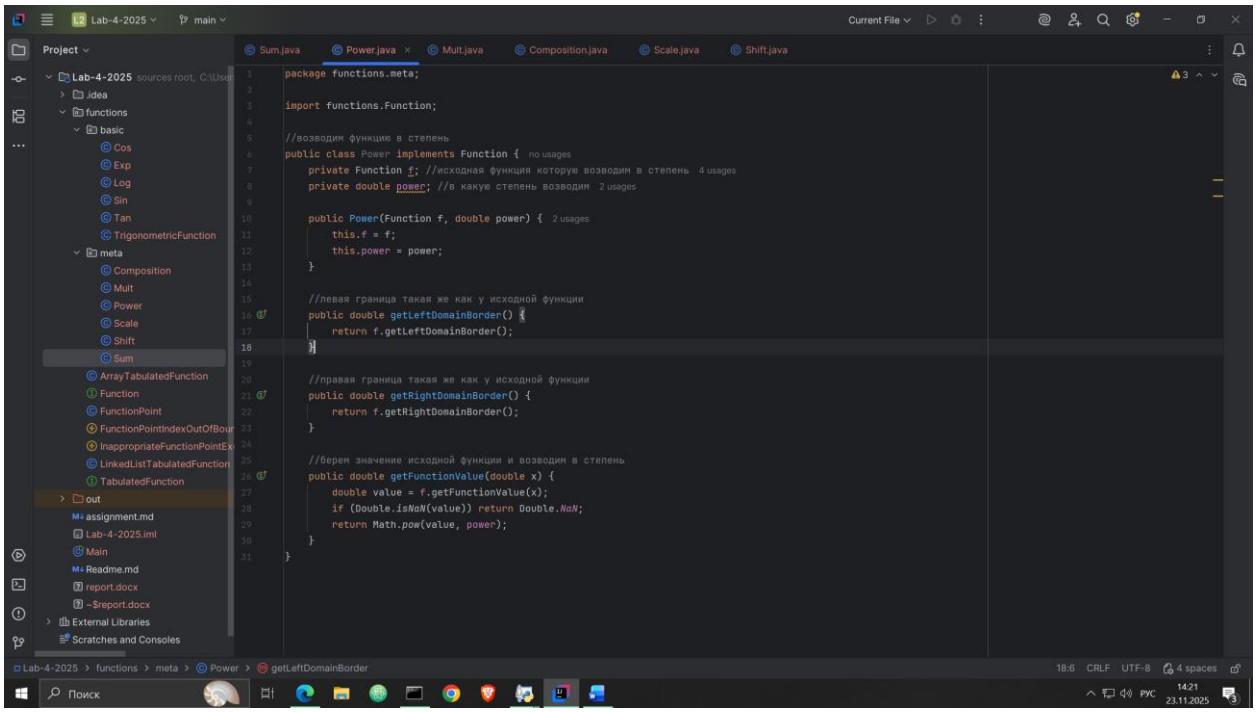
Создан пакет functions.meta

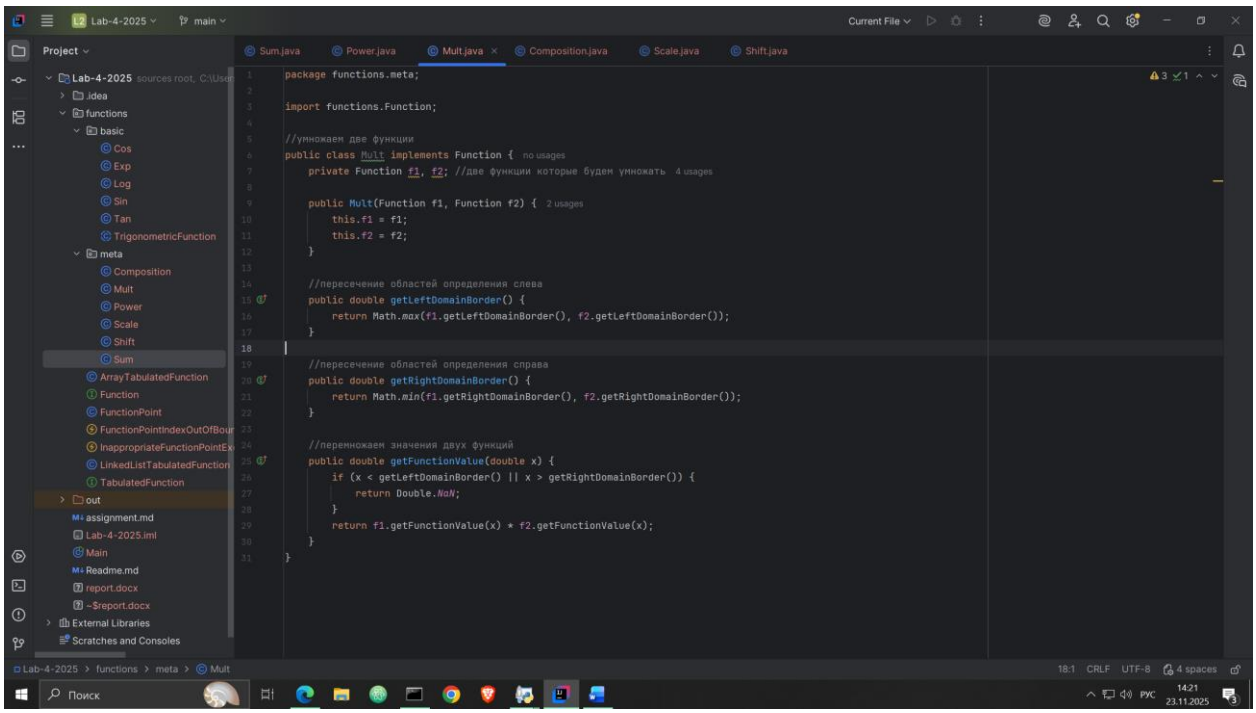
Реализованы классы для комбинирования функций - Sum (сумма двух функций), Mult (произведение двух функций), Power (возведение функции в степень), Scale (масштабирование по осям), Shift (сдвиг по осям), Composition (композиция функций)

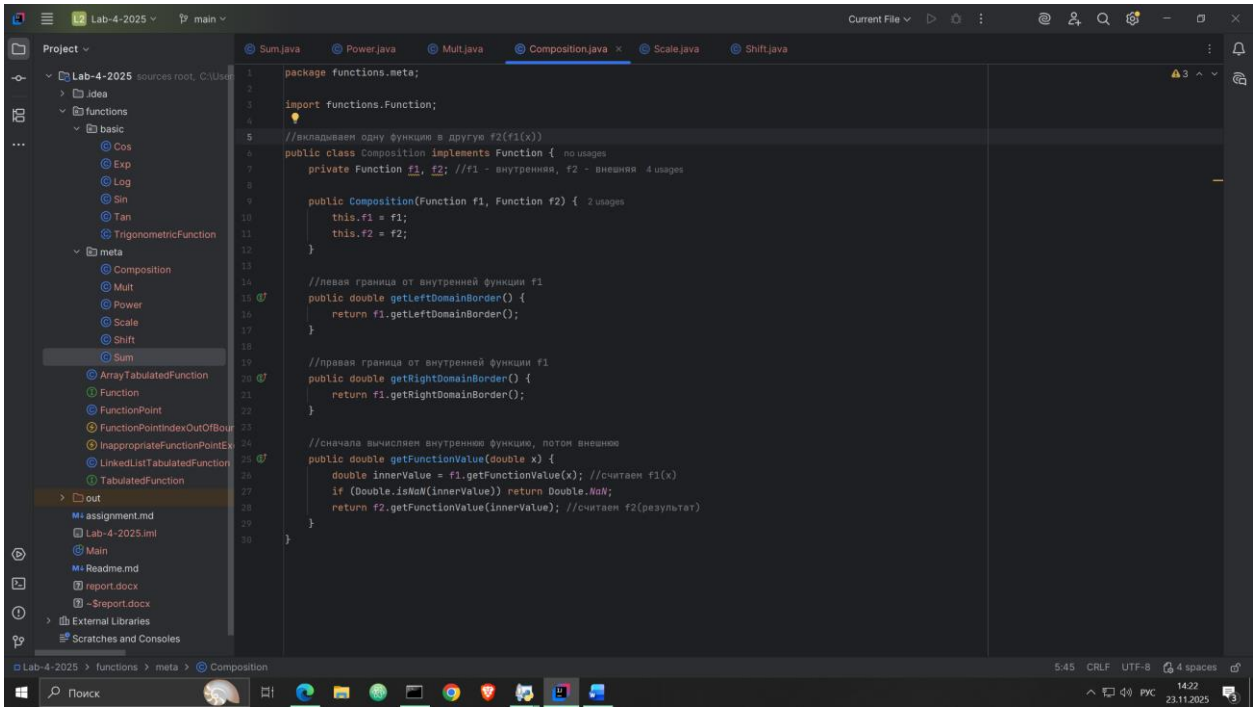
New Package

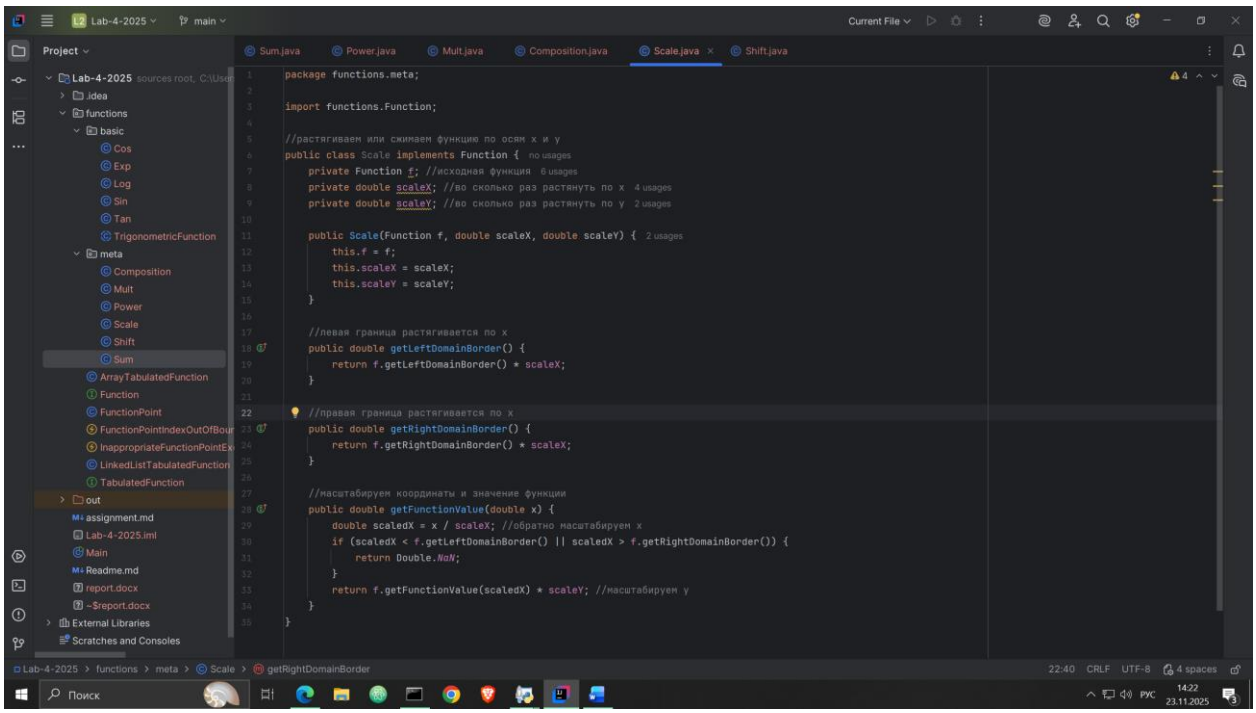
functions.meta

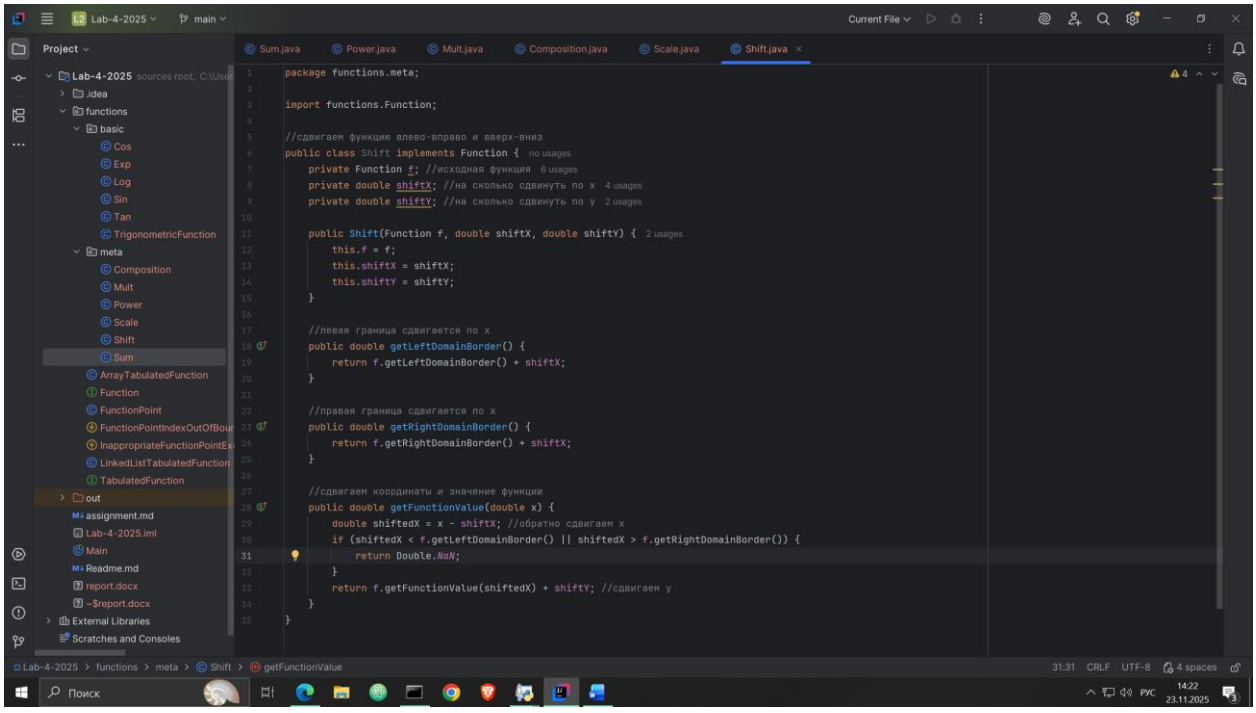





Lab-4-2025
main
Current File
Project
Sum.java
Power.java
Mult.java
Composition.java
Scale.java
Shift.java
Lab-4-2025 sources root, C:\Users
.idea
functions
basic
Cos
Exp
Log
Sin
Tan
TrigonometricFunction
meta
Composition
Mult
Power
Scale
Shift
Sum
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
package functions.meta;
import functions.Function;
//умножаем две функции
public class Mult implements Function { no usages
private Function f1, f2; //две функции которые будем умножать 4 usages
public Mult(Function f1, Function f2) { 2 usages
this.f1 = f1;
this.f2 = f2;
}
//пересечение областей определения слева
public double getLeftDomainBorder() {
return Math.max(f1.getLeftDomainBorder(), f2.getLeftDomainBorder());
}
//пересечение областей определения справа
public double getRightDomainBorder() {
return Math.min(f1.getRightDomainBorder(), f2.getRightDomainBorder());
}
//перемножаем значения двух функций
public double getFunctionValue(double x) {
if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
return Double.NaN;
}
return f1.getFunctionValue(x) * f2.getFunctionValue(x);
}
}
Lab-4-2025 › functions › meta › Mult
18:1 CRLF UTF-8 4 spaces
Поиск
РУС
14:21
23.11.2025


Lab-4-2025
main
Current File
Project
Sum.java
Power.java
Mult.java
Composition.java
Scale.java
Shift.java
Lab-4-2025 sources root, C:\Users
.idea
functions
basic
Cos
Exp
Log
Sin
Tan
TrigonometricFunction
meta
Composition
Mult
Power
Scale
Shift
Sum
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
package functions.meta;
import functions.Function;
//вкладываем одну функцию в другую f2(f1(x))
public class Composition implements Function { no usages
private Function f1, f2; //f1 - внутренняя, f2 - внешняя 4 usages
public Composition(Function f1, Function f2) { 2 usages
this.f1 = f1;
this.f2 = f2;
}
//левая граница от внутренней функции f1
public double getLeftDomainBorder() {
return f1.getLeftDomainBorder();
}
//правая граница от внутренней функции f1
public double getRightDomainBorder() {
return f1.getRightDomainBorder();
}
//сначала вычисляем внутреннюю функцию, потом внешнюю
public double getFunctionValue(double x) {
double innerValue = f1.getFunctionValue(x); //считаем f1(x)
if (Double.isNaN(innerValue)) return Double.NaN;
return f2.getFunctionValue(innerValue); //считаем f2(результат)
}
}
Lab-4-2025 › functions › meta › Composition
5:45 CRLF UTF-8 4 spaces
Поиск
РУС
14:22
23.11.2025

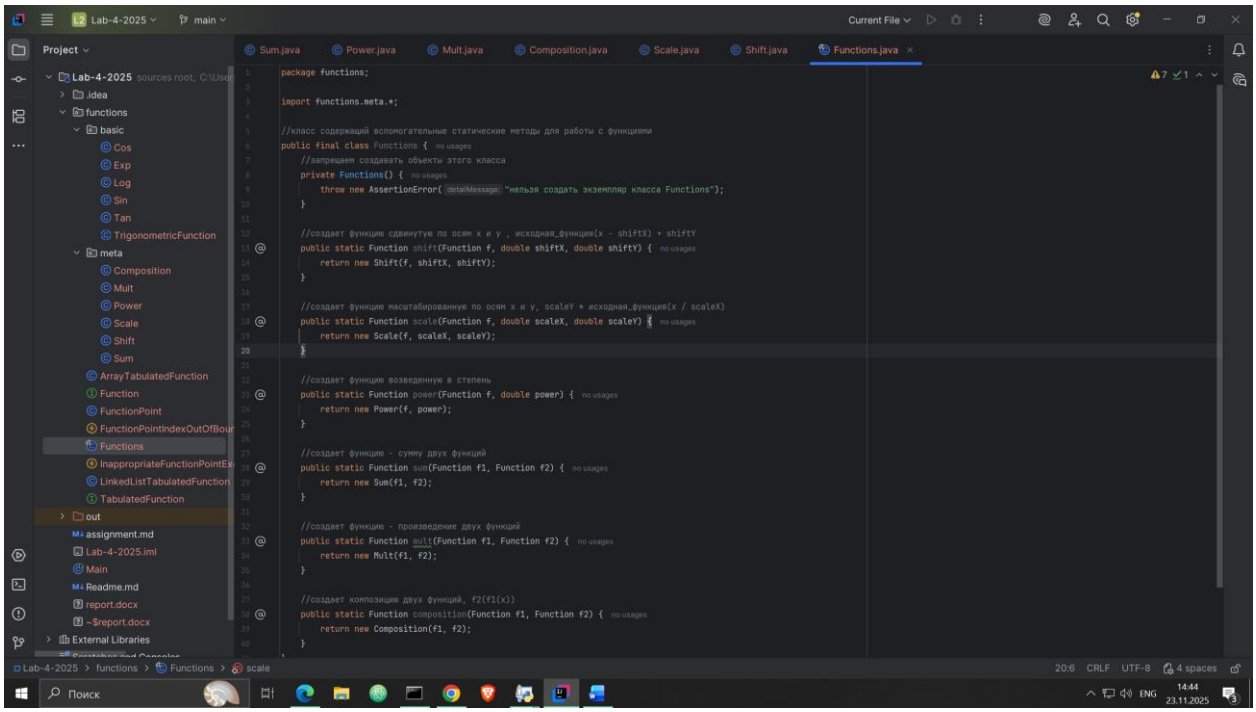

Lab-4-2025
main
Current File
Project
Sum.java
Power.java
Mult.java
Composition.java
Scale.java
Shift.java
Lab-4-2025 sources root, C:\Users
.idea
functions
basic
Cos
Exp
Log
Sin
Tan
TrigonometricFunction
meta
Composition
Mult
Power
Scale
Shift
Sum
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
package functions.meta;
import functions.Function;
//растягиваем или сжимаем функцию по осям x и y
public class Scale implements Function { no usages
private Function f; //исходная функция 6 usages
private double scaleX; //во сколько раз растянуть по x 4 usages
private double scaleY; //во сколько раз растянуть по y 2 usages
public Scale(Function f, double scaleX, double scaleY) { 2 usages
this.f = f;
this.scaleX = scaleX;
this.scaleY = scaleY;
}
//левая граница растягивается по x
public double getLeftDomainBorder() {
return f.getLeftDomainBorder() * scaleX;
}
//правая граница растягивается по x
public double getRightDomainBorder() {
return f.getRightDomainBorder() * scaleX;
}
//масштабируем координаты и значение функции
public double getFunctionValue(double x) {
double scaledX = x / scaleX; //обратно масштабируем x
if (scaledX < f.getLeftDomainBorder() || scaledX > f.getRightDomainBorder()) {
return Double.NaN;
}
return f.getFunctionValue(scaledX) * scaleY; //масштабируем y
}
}
Lab-4-2025 › functions › meta › Scale › getRightDomainBorder
22:40 CRLF UTF-8 4 spaces
Поиск
РУС
14:22
23.11.2025


Lab-4-2025
main
Current File
Project
Sum.java
Power.java
Mult.java
Composition.java
Scale.java
Shift.java
Lab-4-2025 sources root, C:\Users
.idea
functions
basic
Cos
Exp
Log
Sin
Tan
TrigonometricFunction
meta
Composition
Mult
Power
Scale
Shift
Sum
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
package functions.meta;
import functions.Function;
//сдвигаем функцию влево-вправо и вверх-вниз
public class Shift implements Function { no usages
private Function f; //исходная функция 6 usages
private double shiftX; //на сколько сдвинуть по x 4 usages
private double shiftY; //на сколько сдвинуть по y 2 usages
public Shift(Function f, double shiftX, double shiftY) { 2 usages
this.f = f;
this.shiftX = shiftX;
this.shiftY = shiftY;
}
//левая граница сдвигается по x
public double getLeftDomainBorder() {
return f.getLeftDomainBorder() + shiftX;
}
//правая граница сдвигается по x
public double getRightDomainBorder() {
return f.getRightDomainBorder() + shiftX;
}
//сдвигаем координаты и значение функции
public double getFunctionValue(double x) {
double shiftedX = x - shiftX; //обратно сдвигаем x
if (shiftedX < f.getLeftDomainBorder() || shiftedX > f.getRightDomainBorder()) {
return Double.NaN;
}
return f.getFunctionValue(shiftedX) + shiftY; //сдвигаем y
}
}
Lab-4-2025 › functions › meta › Shift › getFunctionValue
31:31 CRLF UTF-8 4 spaces
Поиск
РУС
14:22
23.11.2025

Задание 5

Создан класс Functions со статическими методами-обёртками

Реализованы методы: shift(), scale(), power(), sum(), mult(), composition()

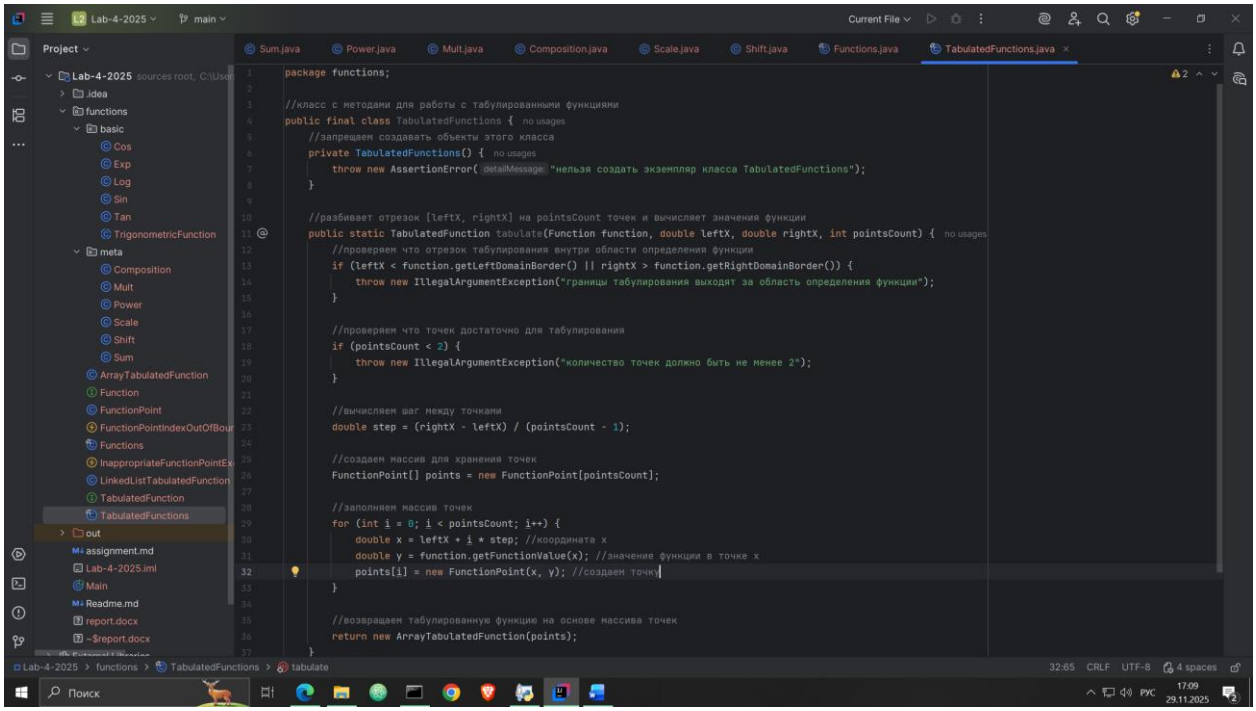


Задание 6

Создан класс TabulatedFunctions со статическими методами

Реализован метод tabulate() для создания табулированных функций из аналитических

Добавлены проверки границ области определения и количества точек



Задание 7

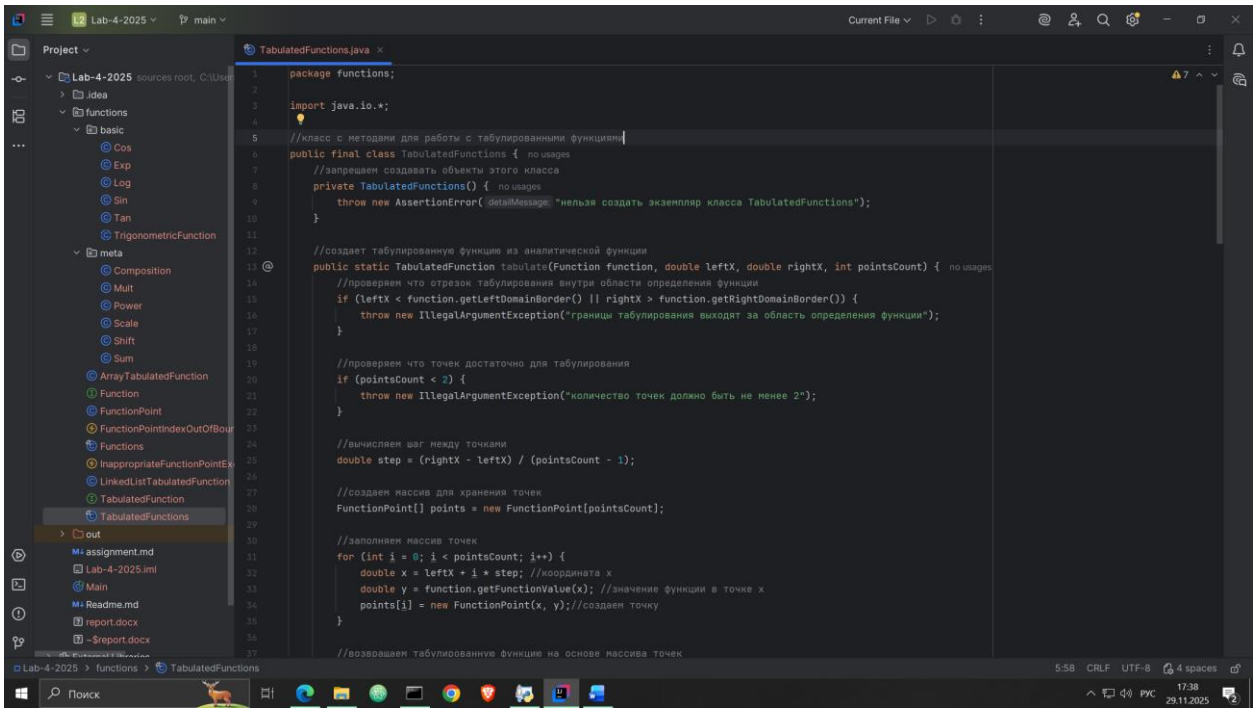
Реализованы методы в классе TabulatedFunctions

outputTabulatedFunction() - запись в байтовый поток

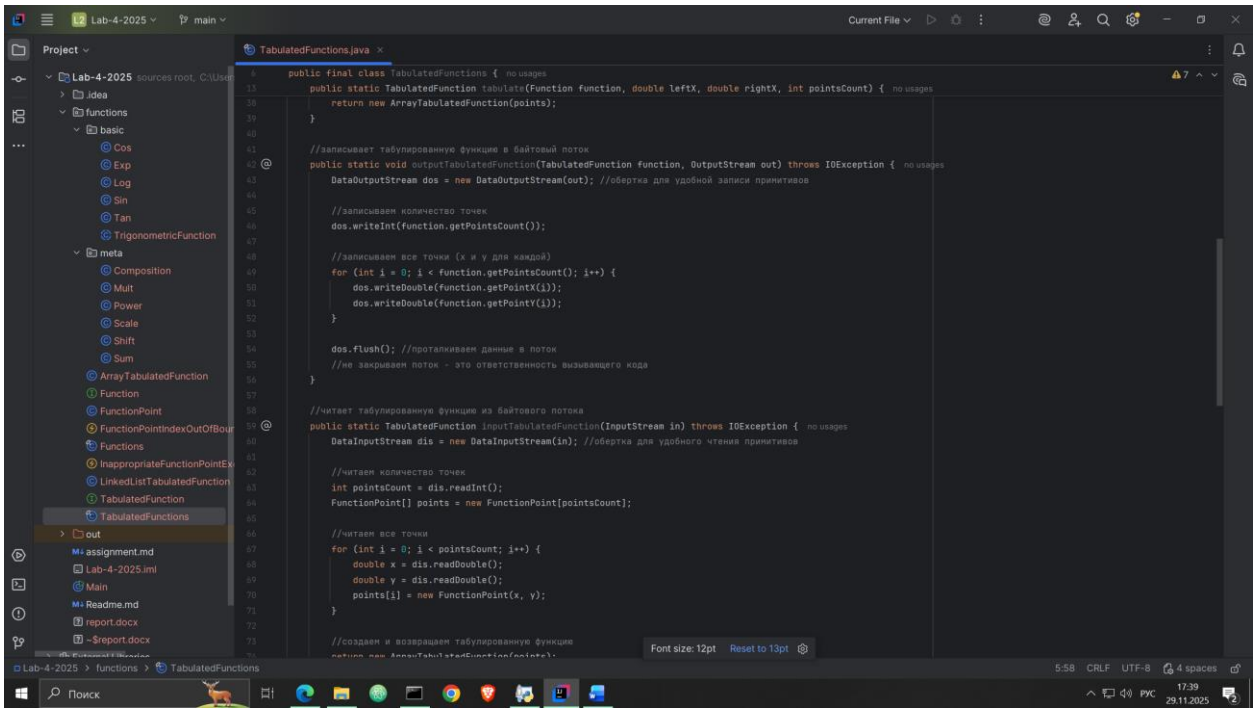
inputTabulatedFunction() - чтение из байтового потока

writeTabulatedFunction() - запись в символьный поток

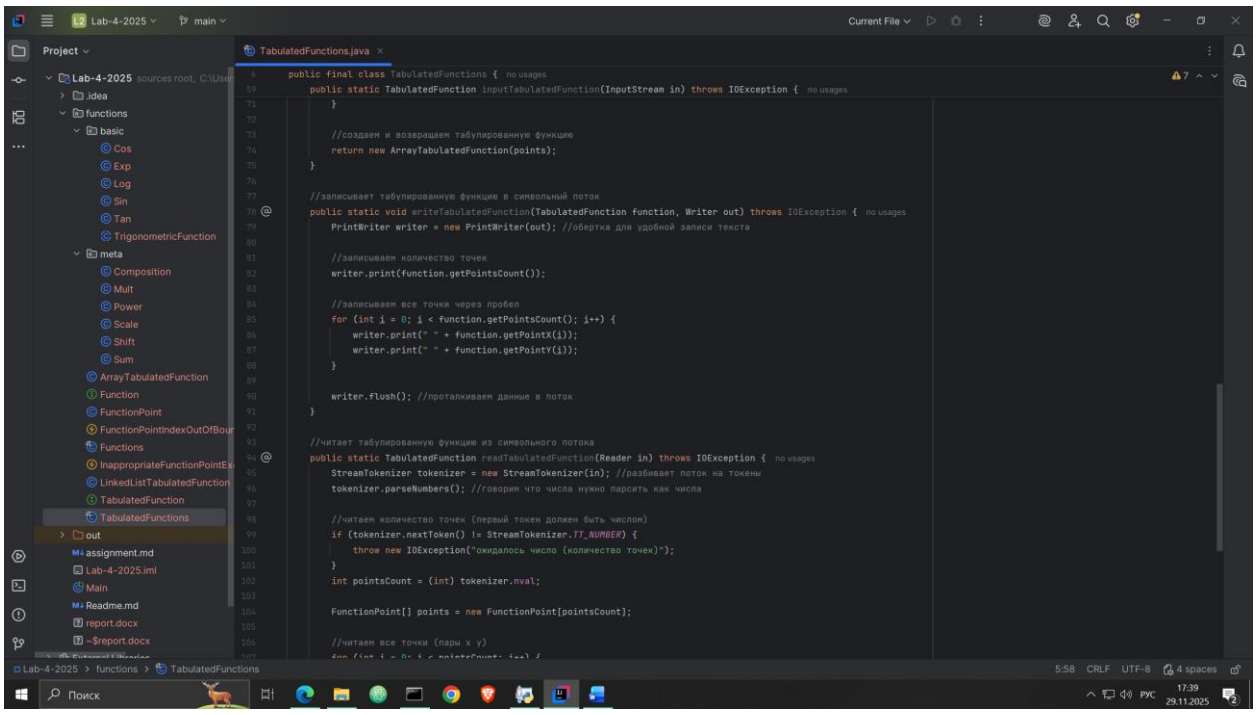
readTabulatedFunction() - чтение из символьного потока

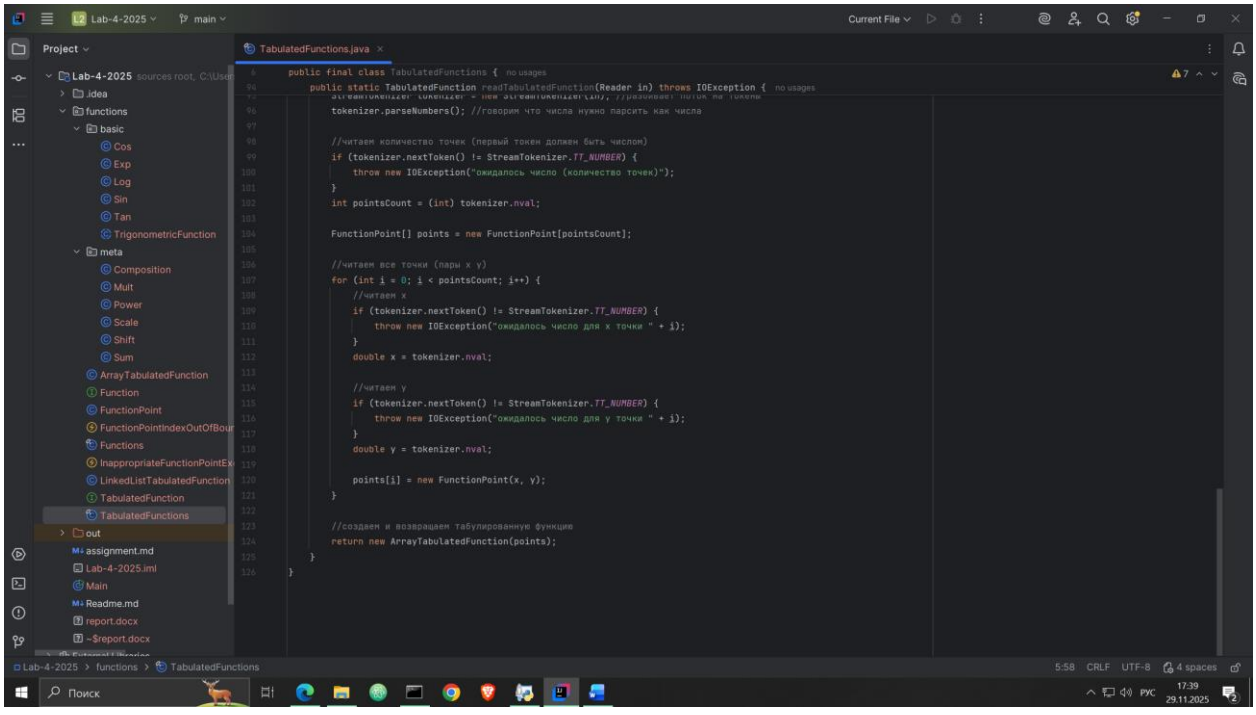


```
package functions;

import java.io.*;

//класс с методами для работы с табулированными функциями
public final class TabulatedFunctions {
    //запрещаем создавать объекты этого класса
    private TabulatedFunctions() {
        throw new AssertionError("нельзя создать экземпляр класса TabulatedFunctions");
    }

    //создает табулированную функцию из аналитической функции
    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
        //проверяем что отрезок табулирования внутри области определения функции
        if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
            throw new IllegalArgumentException("границы табулирования выходят за область определения функции");
        }

        //проверяем что точек достаточно для табулирования
        if (pointsCount < 2) {
            throw new IllegalArgumentException("количество точек должно быть не менее 2");
        }

        //вычисляем шаг между точками
        double step = (rightX - leftX) / (pointsCount - 1);

        //создаем массив для хранения точек
        FunctionPoint[] points = new FunctionPoint[pointsCount];

        //заполняем массив точек
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step; //координата x
            double y = function.getFunctionValue(x); //значение функции в точке x
            points[i] = new FunctionPoint(x, y);//создаем точку
        }

        //возвращаем табулированную функцию на основе массива точек
```

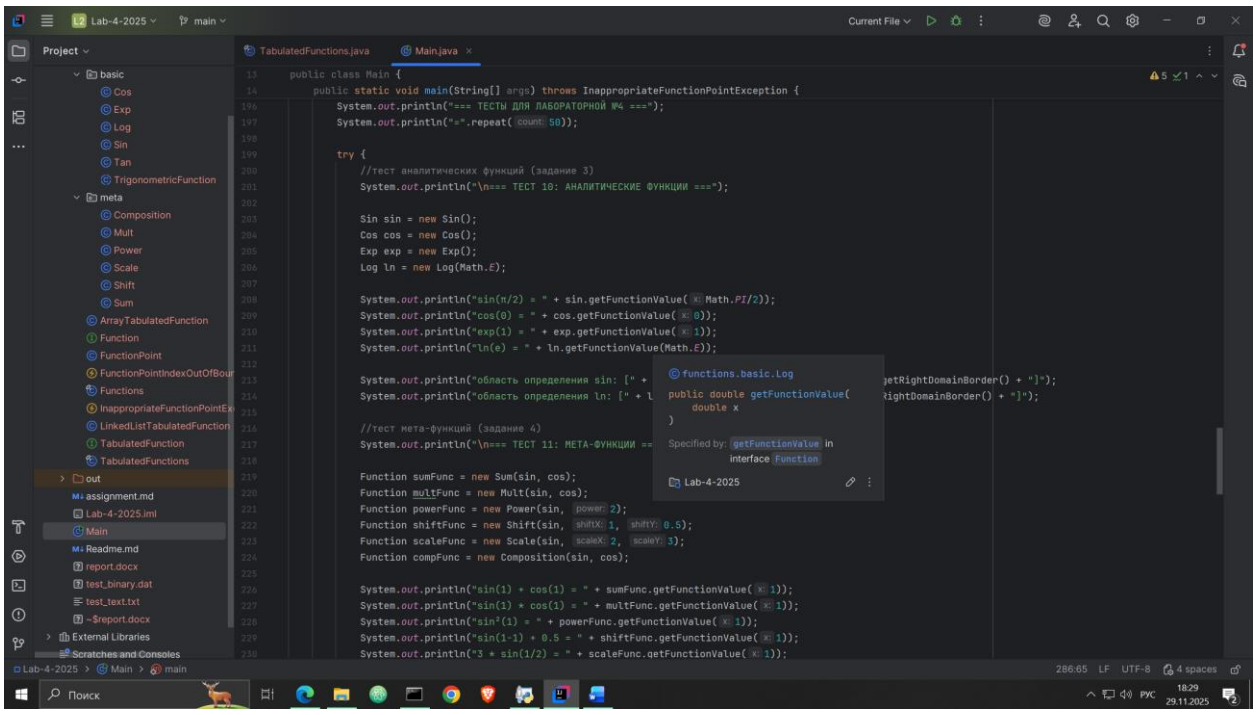


```
public final class TabulatedFunctions {
    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
        return new ArrayTabulatedFunction(points);
    }

    //записывает табулированную функцию в байтовый поток
    public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException {
        DataOutputStream dos = new DataOutputStream(out); //обертка для удобной записи примитивов

        //записываем количество точек
        dos.writeInt(function.getPointsCount());

        //записываем все точки (x и y для каждой)
        for (int i = 0; i < function.getPointsCount(); i++) {
            dos.writeDouble(function.getPointX(i));
            dos.writeDouble(function.getPointY(i));
        }

        dos.flush(); //проталкиваем данные в поток
        //не закрываем поток - это ответственность вызывающего кода
    }

    //читает табулированную функцию из байтового потока
    public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException {
        DataInputStream dis = new DataInputStream(in); //обертка для удобного чтения примитивов

        //читаем количество точек
        int pointsCount = dis.readInt();
        FunctionPoint[] points = new FunctionPoint[pointsCount];

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

        //создаем и возвращаем табулированную функцию
```


Lab-4-2025
main
Current File
Project
TabulatedFunctions.java
Lab-4-2025 sources root, C:\Users
.idea
functions
basic
Cos
Exp
Log
Sin
Tan
TrigonometricFunction
meta
Composition
Mult
Power
Scale
Shift
Sum
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
Functions
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
TabulatedFunctions
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
public final class TabulatedFunctions { no usages
public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException { no usages
}
//создаем и возвращаем табулированную функцию
return new ArrayTabulatedFunction(points);
}
//записывает табулированную функцию в символьный поток
public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException { no usages
PrintWriter writer = new PrintWriter(out); //обертка для удобной записи текста
//записываем количество точек
writer.print(function.getPointsCount());
//записываем все точки через пробел
for (int i = 0; i < function.getPointsCount(); i++) {
writer.print(" " + function.getPointX(i));
writer.print(" " + function.getPointY(i));
}
writer.flush(); //проталкиваем данные в поток
}
//читает табулированную функцию из символьного потока
public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException { no usages
StreamTokenizer tokenizer = new StreamTokenizer(in); //разбивает поток на токены
tokenizer.parseNumbers(); //говорим что числа нужно парсить как числа
//читаем количество точек (первый токен должен быть числом)
if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
throw new IOException("ожидалось число (количество точек)");
}
int pointsCount = (int) tokenizer.nval;
FunctionPoint[] points = new FunctionPoint[pointsCount];
//читаем все точки (пары x y)
Lab-4-2025 › functions › TabulatedFunctions
5:58 CRLF UTF-8 4 spaces
Поиск
РУС
17:39
29.11.2025

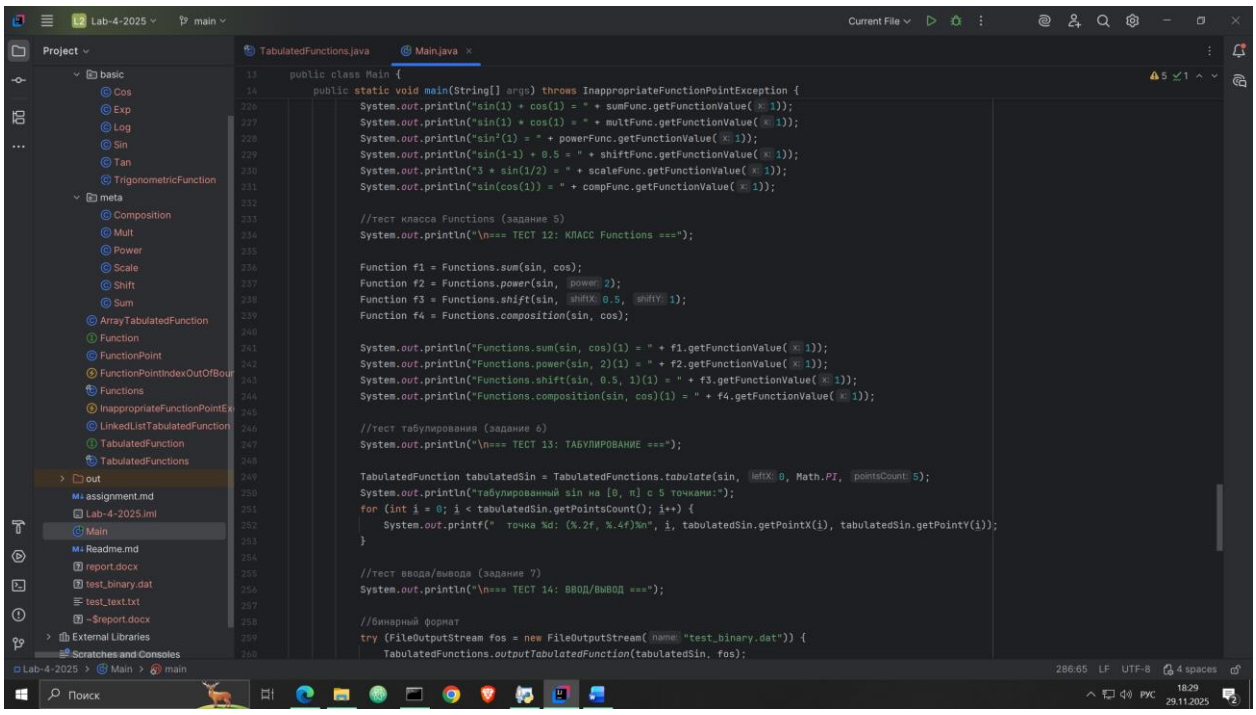

Lab-4-2025
main
Current File
Project
TabulatedFunctions.java
Lab-4-2025 sources root, C:\Users
.idea
functions
basic
Cos
Exp
Log
Sin
Tan
TrigonometricFunction
meta
Composition
Mult
Power
Scale
Shift
Sum
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBour
Functions
InappropriateFunctionPointEx
LinkedListTabulatedFunction
TabulatedFunction
TabulatedFunctions
out
assignment.md
Lab-4-2025.iml
Main
Readme.md
report.docx
~$report.docx
public final class TabulatedFunctions { no usages
public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException { no usages
tokenizer.parseNumbers(); //говорим что числа нужно парсить как числа
//читаем количество точек (первый токен должен быть числом)
if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
throw new IOException("ожидалось число (количество точек)");
}
int pointsCount = (int) tokenizer.nval;
FunctionPoint[] points = new FunctionPoint[pointsCount];
//читаем все точки (пары x y)
for (int i = 0; i < pointsCount; i++) {
//читаем x
if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
throw new IOException("ожидалось число для x точки " + i);
}
double x = tokenizer.nval;
//читаем y
if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
throw new IOException("ожидалось число для y точки " + i);
}
double y = tokenizer.nval;
points[i] = new FunctionPoint(x, y);
}
//создаем и возвращаем табулированную функцию
return new ArrayTabulatedFunction(points);
}
}
Lab-4-2025 › functions › TabulatedFunctions
5:58 CRLF UTF-8 4 spaces
Поиск
РУС
17:39
29.11.2025

Задание 8

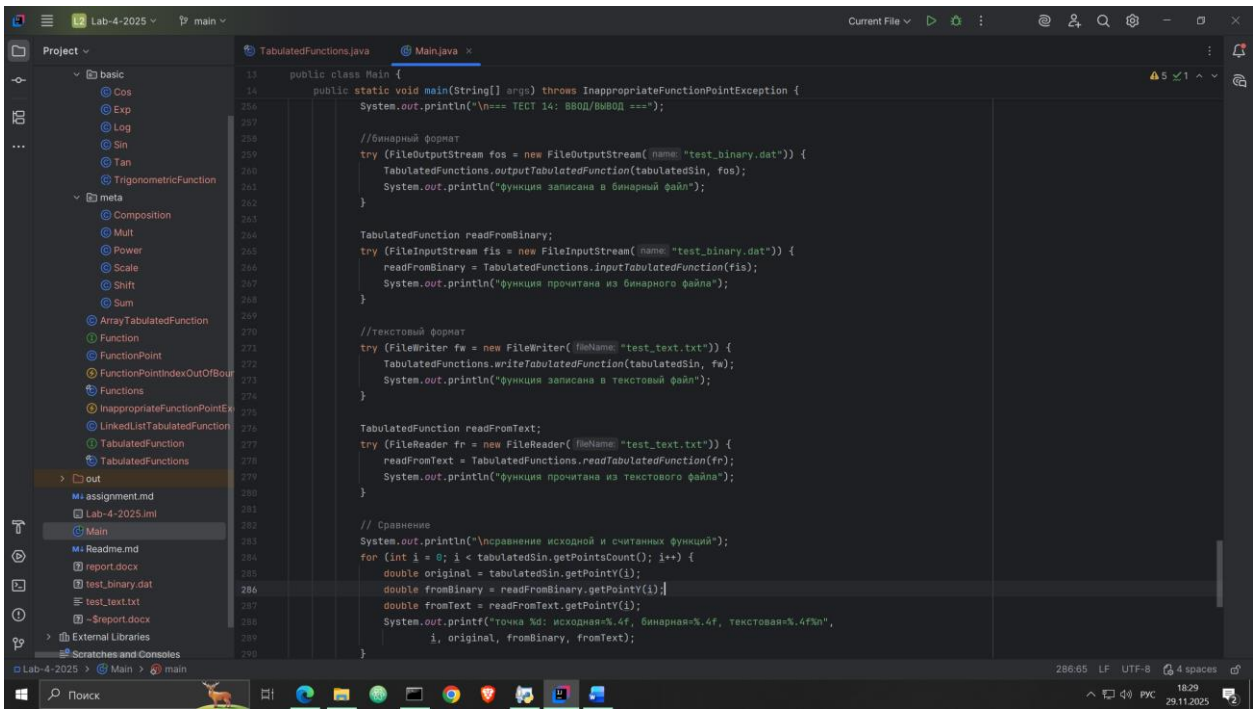


```
public class Main {
    public static void main(String[] args) throws InappropriateFunctionPointException {
        System.out.println("=== ТЕСТЫ ДЛЯ ЛАБОРАТОРНОЙ №4 ===");
        System.out.println("=".repeat( count: 50));

        try {
            //тест аналитических функций (задание 3)
            System.out.println("\n=== ТЕСТ 10: АНАЛИТИЧЕСКИЕ ФУНКЦИИ ===");

            Sin sin = new Sin();
            Cos cos = new Cos();
            Exp exp = new Exp();
            Log ln = new Log(Math.E);

            System.out.println("sin(π/2) = " + sin.getFunctionValue( x: Math.PI/2));
            System.out.println("cos(0) = " + cos.getFunctionValue( x: 0));
            System.out.println("exp(1) = " + exp.getFunctionValue( x: 1));
            System.out.println("ln(e) = " + ln.getFunctionValue(Math.E));

            System.out.println("область определения sin: [" +                    getRightDomainBorder() + "]");
            System.out.println("область определения ln: [" + L                   RightDomainBorder() + "]");

            //тест мета-функций (задание 4)
            System.out.println("\n=== ТЕСТ 11: МЕТА-ФУНКЦИИ ==

            Function sumFunc = new Sum(sin, cos);
            Function multFunc = new Mult(sin, cos);
            Function powerFunc = new Power(sin, power: 2);
            Function shiftFunc = new Shift(sin, shiftX: 1, shiftY: 0.5);
            Function scaleFunc = new Scale(sin, scaleX: 2, scaleY: 3);
            Function compFunc = new Composition(sin, cos);

            System.out.println("sin(1) + cos(1) = " + sumFunc.getFunctionValue( x: 1));
            System.out.println("sin(1) * cos(1) = " + multFunc.getFunctionValue( x: 1));
            System.out.println("sin²(1) = " + powerFunc.getFunctionValue( x: 1));
            System.out.println("sin(1-1) + 0.5 = " + shiftFunc.getFunctionValue( x: 1));
            System.out.println("3 * sin(1/2) = " + scaleFunc.getFunctionValue( x: 1));
```

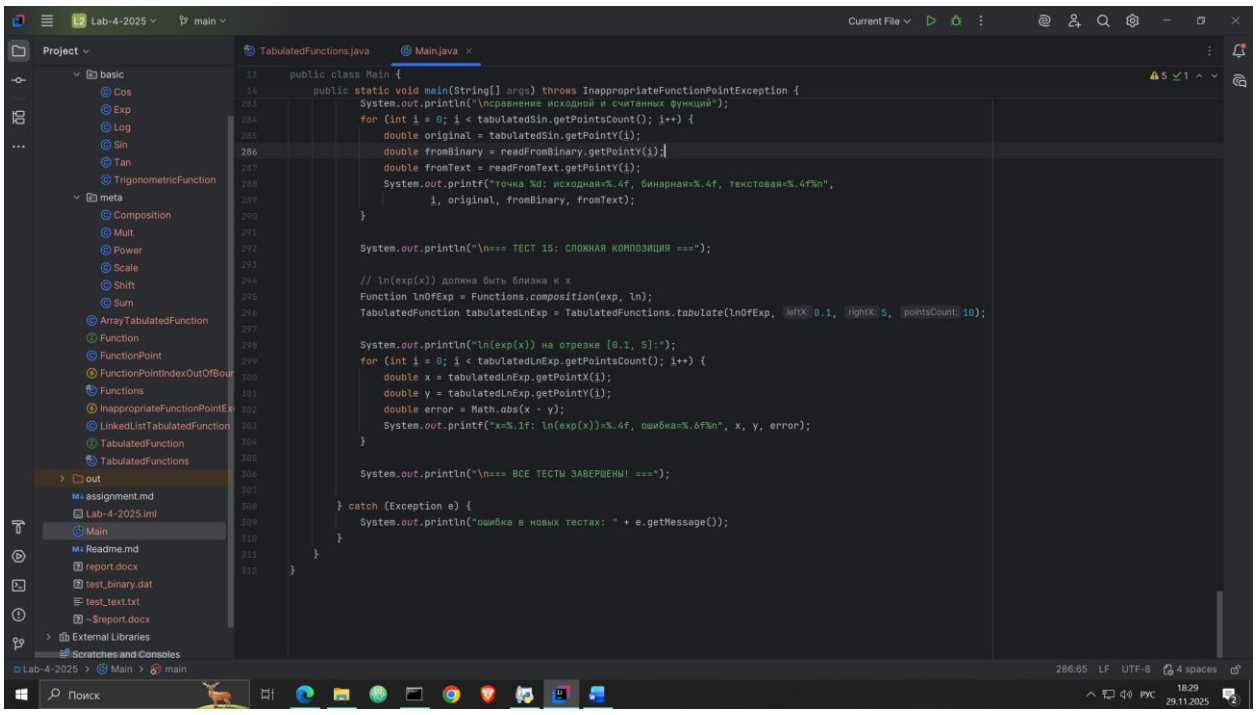
functions.basic.Log
public double getFunctionValue(
double x
)
Specified by: getFunctionValue in interface Function
Lab-4-2025



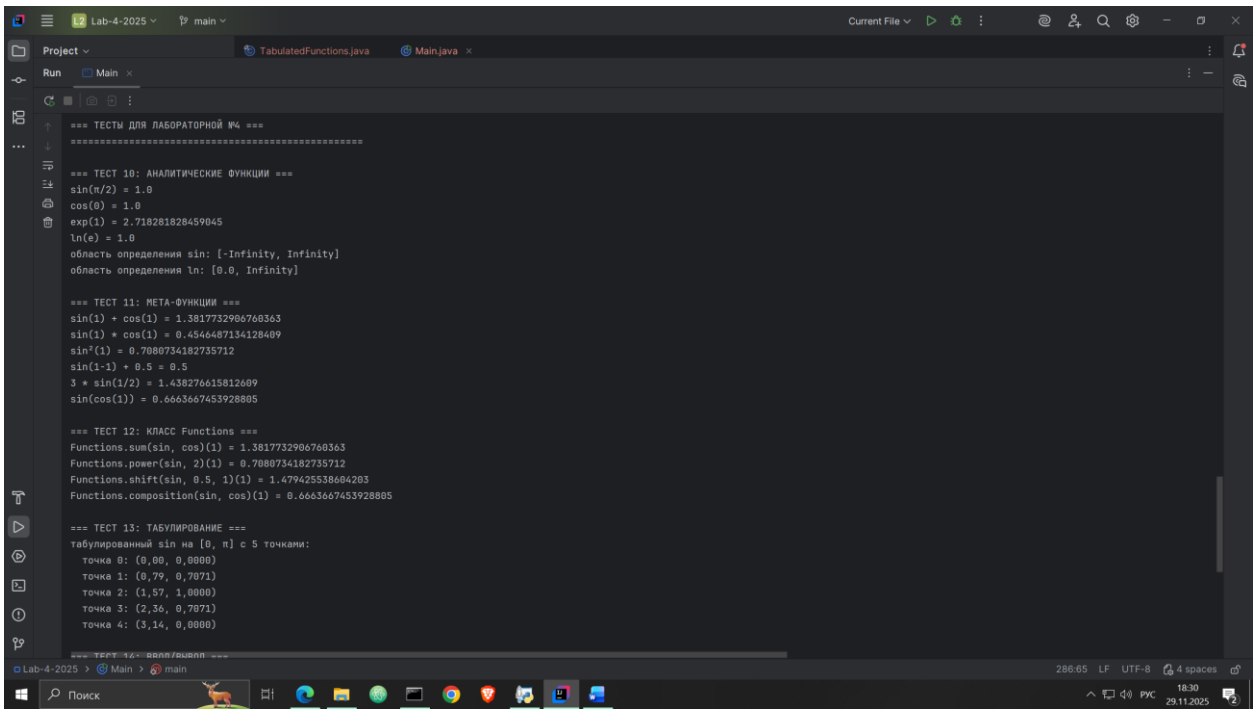
```
public class Main {
    public static void main(String[] args) throws InappropriateFunctionPointException {
            System.out.println("sin(1) + cos(1) = " + sumFunc.getFunctionValue( x: 1));
            System.out.println("sin(1) * cos(1) = " + multFunc.getFunctionValue( x: 1));
            System.out.println("sin²(1) = " + powerFunc.getFunctionValue( x: 1));
            System.out.println("sin(1-1) + 0.5 = " + shiftFunc.getFunctionValue( x: 1));
            System.out.println("3 * sin(1/2) = " + scaleFunc.getFunctionValue( x: 1));
            System.out.println("sin(cos(1)) = " + compFunc.getFunctionValue( x: 1));

            //тест класса Functions (задание 5)
            System.out.println("\n=== ТЕСТ 12: КЛАСС Functions ===");

            Function f1 = Functions.sum(sin, cos);
            Function f2 = Functions.power(sin, power: 2);
            Function f3 = Functions.shift(sin, shiftX: 0.5, shiftY: 1);
            Function f4 = Functions.composition(sin, cos);

            System.out.println("Functions.sum(sin, cos)(1) = " + f1.getFunctionValue( x: 1));
            System.out.println("Functions.power(sin, 2)(1) = " + f2.getFunctionValue( x: 1));
            System.out.println("Functions.shift(sin, 0.5, 1)(1) = " + f3.getFunctionValue( x: 1));
            System.out.println("Functions.composition(sin, cos)(1) = " + f4.getFunctionValue( x: 1));

            //тест табулирования (задание 6)
            System.out.println("\n=== ТЕСТ 13: ТАБУЛИРОВАНИЕ ===");

            TabulatedFunction tabulatedSin = TabulatedFunctions.tabulate(sin, leftX: 0, Math.PI, pointsCount: 5);
            System.out.println("Табулированный sin на [0, π] с 5 точками:");
            for (int i = 0; i < tabulatedSin.getPointsCount(); i++) {
                System.out.printf("  точка %d: (%.2f, %.4f)%n", i, tabulatedSin.getPointX(i), tabulatedSin.getPointY(i));
            }

            //тест ввода/вывода (задание 7)
            System.out.println("\n=== ТЕСТ 14: ВВОД/ВЫВОД ===");

            //бинарный формат
            try (FileOutputStream fos = new FileOutputStream( name: "test_binary.dat")) {
                TabulatedFunctions.outputTabulatedFunction(tabulatedSin, fos);
```

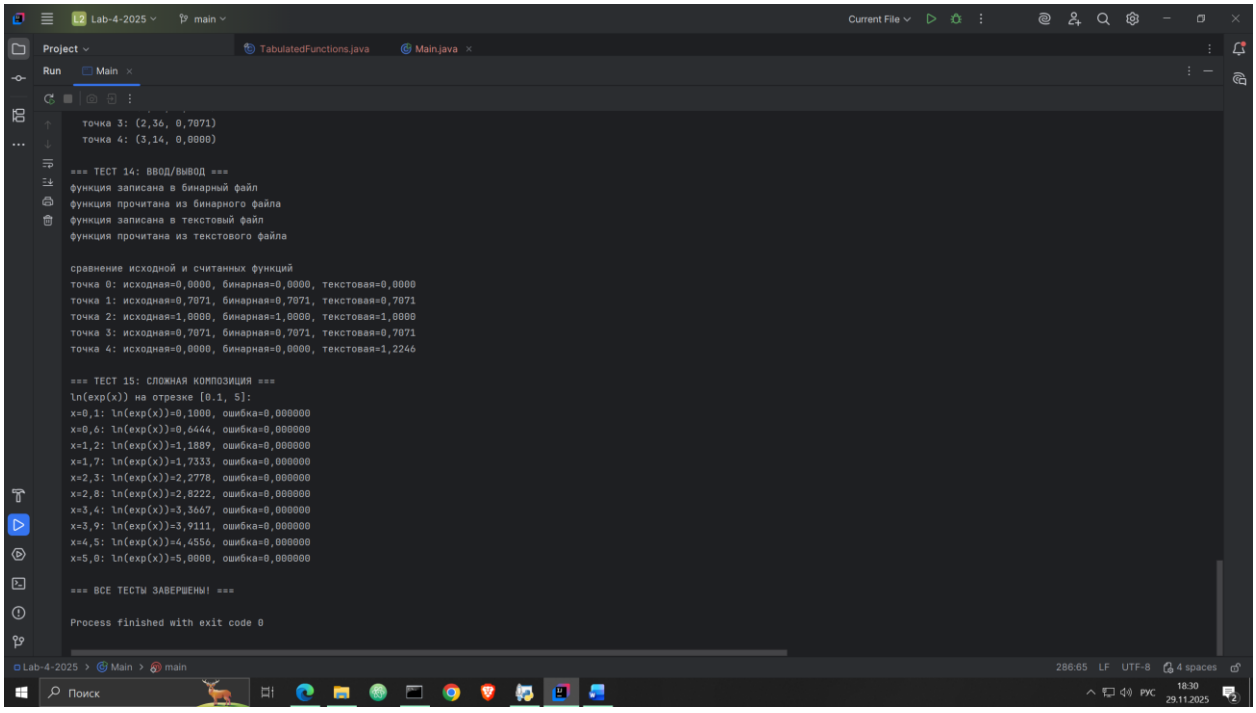


```
public class Main {
    public static void main(String[] args) throws InappropriateFunctionPointException {
        System.out.println("\n=== ТЕСТ 14: ВВОД/ВЫВОД ===");

        //бинарный формат
        try (FileOutputStream fos = new FileOutputStream( name: "test_binary.dat")) {
            TabulatedFunctions.outputTabulatedFunction(tabulatedSin, fos);
            System.out.println("функция записана в бинарный файл");
        }

        TabulatedFunction readFromBinary;
        try (FileInputStream fis = new FileInputStream( name: "test_binary.dat")) {
            readFromBinary = TabulatedFunctions.inputTabulatedFunction(fis);
            System.out.println("функция прочитана из бинарного файла");
        }

        //текстовый формат
        try (FileWriter fw = new FileWriter( fileName: "test_text.txt")) {
            TabulatedFunctions.writeTabulatedFunction(tabulatedSin, fw);
            System.out.println("функция записана в текстовый файл");
        }

        TabulatedFunction readFromText;
        try (FileReader fr = new FileReader( fileName: "test_text.txt")) {
            readFromText = TabulatedFunctions.readTabulatedFunction(fr);
            System.out.println("функция прочитана из текстового файла");
        }

        // Сравнение
        System.out.println("\nсравнение исходной и считанных функций");
        for (int i = 0; i < tabulatedSin.getPointsCount(); i++) {
            double original = tabulatedSin.getPointY(i);
            double fromBinary = readFromBinary.getPointY(i);
            double fromText = readFromText.getPointY(i);
            System.out.printf("точка %d: исходная=%.4f, бинарная=%.4f, текстовая=%.4f%n",
                    i, original, fromBinary, fromText);
        }
```



```
public class Main {
    public static void main(String[] args) throws InappropriateFunctionPointException {
        System.out.println("\nсравнение исходной и считанных функций");
        for (int i = 0; i < tabulatedSin.getPointsCount(); i++) {
            double original = tabulatedSin.getPointY(i);
            double fromBinary = readFromBinary.getPointY(i);
            double fromText = readFromText.getPointY(i);
            System.out.printf("точка %d: исходная=%.4f, бинарная=%.4f, текстовая=%.4f%n",
                    i, original, fromBinary, fromText);
        }

        System.out.println("\n=== ТЕСТ 15: СЛОЖНАЯ КОМПОЗИЦИЯ ===");

        // ln(exp(x)) должна быть близка к x
        Function lnOfExp = Functions.composition(exp, ln);
        TabulatedFunction tabulatedLnExp = TabulatedFunctions.tabulate(lnOfExp, leftX: 0.1, rightX: 5, pointsCount: 10);

        System.out.println("ln(exp(x)) на отрезке [0.1, 5]:");
        for (int i = 0; i < tabulatedLnExp.getPointsCount(); i++) {
            double x = tabulatedLnExp.getPointX(i);
            double y = tabulatedLnExp.getPointY(i);
            double error = Math.abs(x - y);
            System.out.printf("x=%.1f: ln(exp(x))=%.4f, ошибка=%.6f%n", x, y, error);
        }

        System.out.println("\n=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ! ===");

    } catch (Exception e) {
        System.out.println("ошибка в новых тестах: " + e.getMessage());
    }
}
}
```

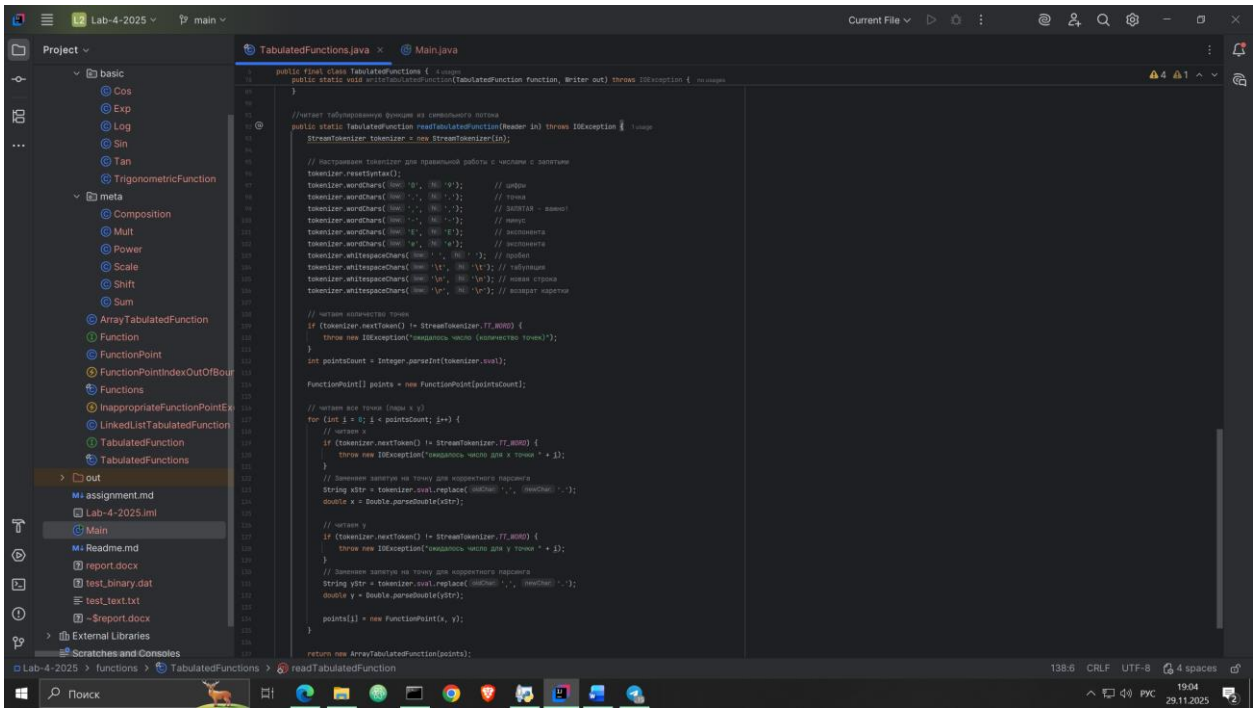


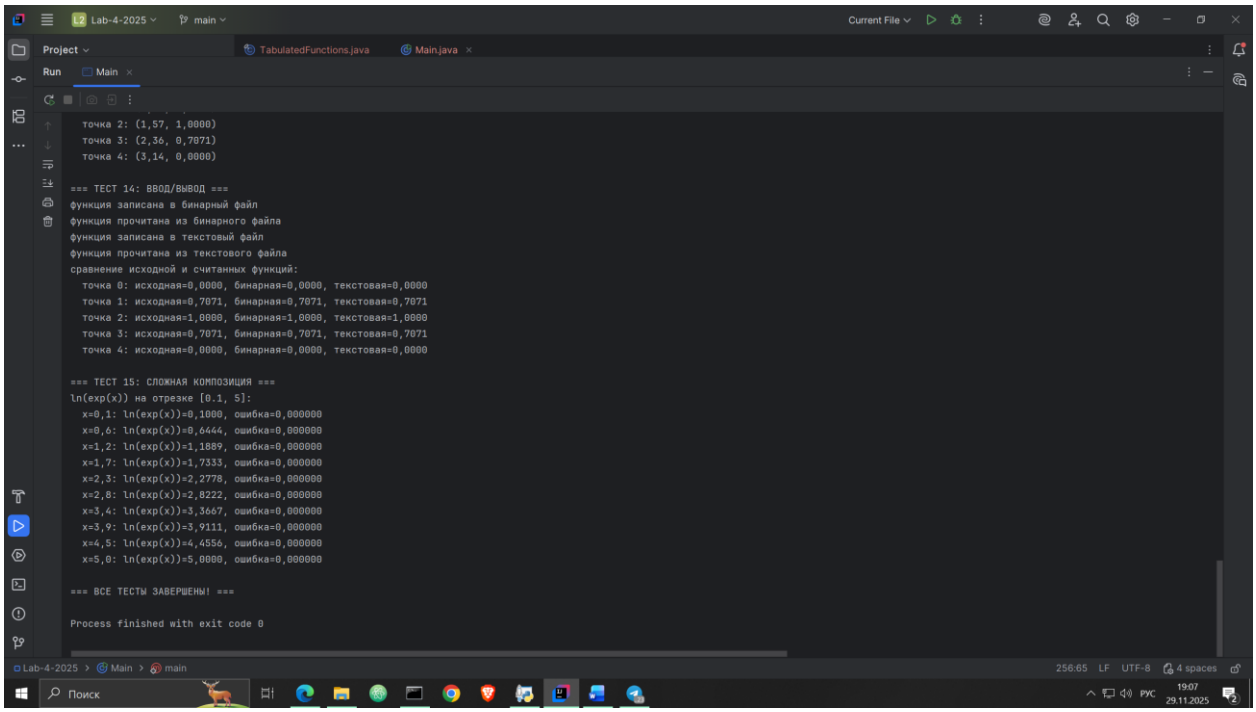


При тестировании выявились проблемы

Проблема : точка 4: текстовая=1,2246 вместо 0,0000, StreamTokenizer неправильно парсит числа с запятыми в дробной части

Решение проблемы





Задание 9

Реализована сериализация с использованием Serializable

Добавлен implements Serializable в классы – FunctionPoint, ArrayTabulatedFunction, LinkedListTabulatedFunction

Протестирована сериализация и десериализация функций

TabulatedFunctions.java Main.java FunctionPoint.java ×

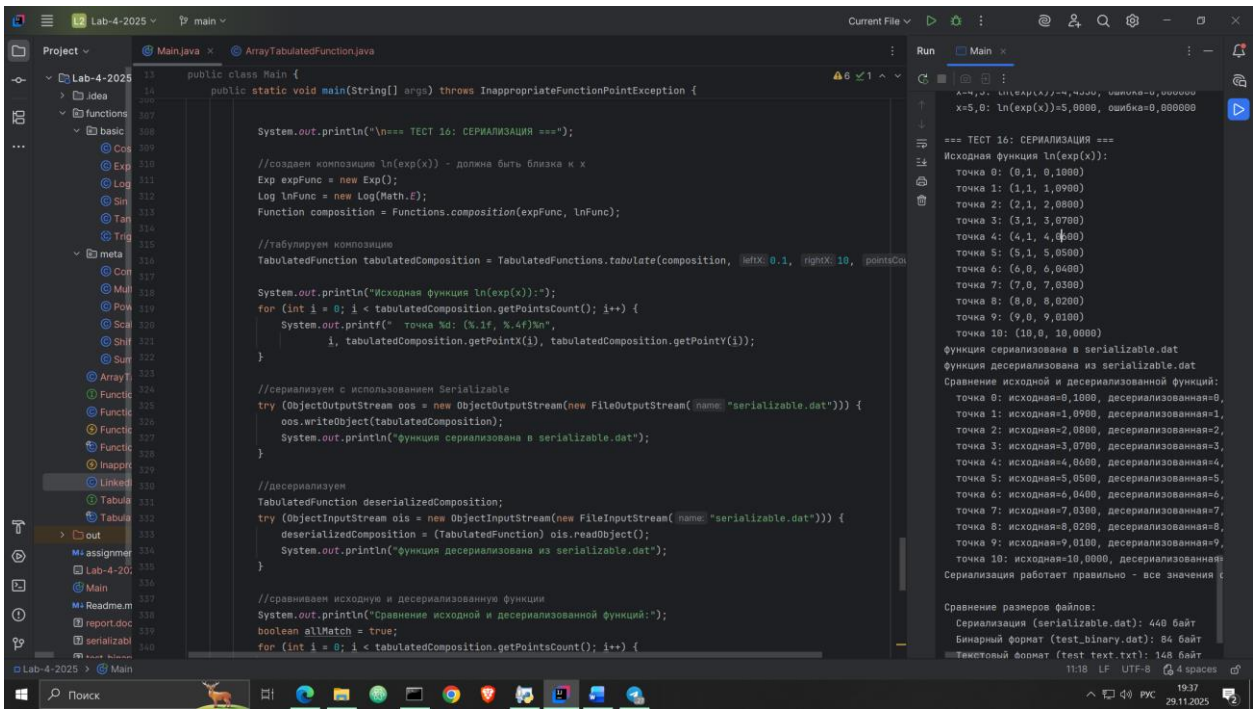
```
1   package functions;
2
3   import java.io.Serializable;
4
5   public class FunctionPoint implements Serializable {  50 usages
6       private static final long serialVersionUID = 1L;  no usages
7       private double x;  5 usages
8       private double y;  5 usages
9
10      //создаёт объект точки с заданными координатами
```

TabulatedFunctions.java Main.java FunctionPoint.java ArrayTabulatedFunction.java ×

```
1   package functions;
2
3   import java.io.Serializable;
4
5
6   public class ArrayTabulatedFunction implements TabulatedFunction, Serializable {
7       private static final long serialVersionUID = 1L;  7 usages
8
9       private FunctionPoint[] points; //массив типа FunctionPoint  36 usages
```

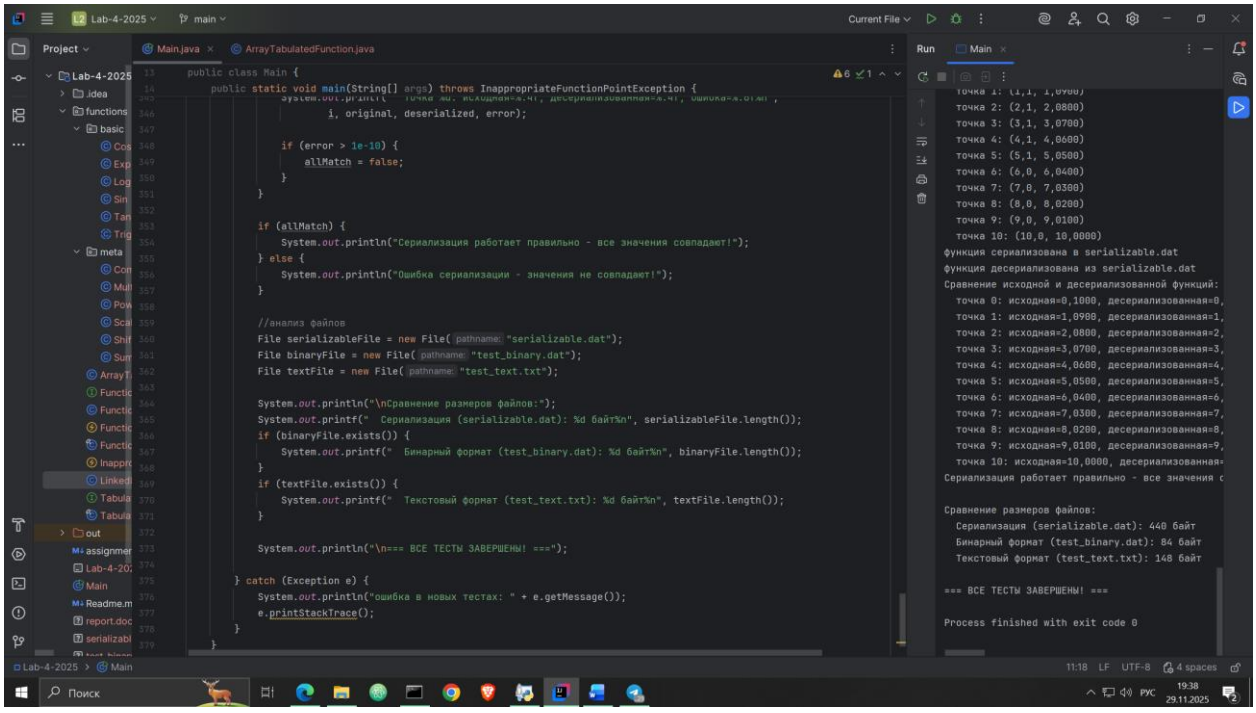
TabulatedFunctions.java × Main.java FunctionPoint.java ArrayTabulatedFunction.java LinkedListTabulatedFunction.java ×

```
1   package functions;
2   import java.io.Serializable;
3   public class LinkedListTabulatedFunction implements TabulatedFunction, Serializable {  4 usages
4       private static final long serialVersionUID = 1L;  no usages
5
6       private static class FunctionNode implements Serializable {  25 usages
7           private static final long serialVersionUID = 1L;  no usages
8           FunctionPoint point;  21 usages
9           FunctionNode prev;  13 usages
10          FunctionNode next;  17 usages
11
12          //конструктор узла
```



```
System.out.println("\n=== ТЕСТ 16: СЕРИАЛИЗАЦИЯ ===");

//создаем композицию ln(exp(x)) - должна быть близка к x
Exp expFunc = new Exp();
Log lnFunc = new Log(Math.E);
Function composition = Functions.composition(expFunc, lnFunc);

//табулируем композицию
TabulatedFunction tabulatedComposition = TabulatedFunctions.tabulate(composition, leftX: 0.1, rightX: 10, pointsCo

System.out.println("Исходная функция ln(exp(x)):");
for (int i = 0; i < tabulatedComposition.getPointsCount(); i++) {
    System.out.printf("  точка %d: (%.1f, %.4f)%n",
            i, tabulatedComposition.getPointX(i), tabulatedComposition.getPointY(i));
}

//сериализуем с использованием Serializable
try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream( name: "serializable.dat"))) {
    oos.writeObject(tabulatedComposition);
    System.out.println("функция сериализована в serializable.dat");
}

//десериализуем
TabulatedFunction deserializedComposition;
try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream( name: "serializable.dat"))) {
    deserializedComposition = (TabulatedFunction) ois.readObject();
    System.out.println("функция десериализована из serializable.dat");
}

//сравниваем исходную и десериализованную функции
System.out.println("Сравнение исходной и десериализованной функций:");
boolean allMatch = true;
for (int i = 0; i < tabulatedComposition.getPointsCount(); i++) {
```

```
x=5,0: ln(exp(x))=5,0000, ошибка=0,000000

=== ТЕСТ 16: СЕРИАЛИЗАЦИЯ ===
Исходная функция ln(exp(x)):
  точка 0: (0,1, 0,1000)
  точка 1: (1,1, 1,0900)
  точка 2: (2,1, 2,0800)
  точка 3: (3,1, 3,0700)
  точка 4: (4,1, 4,0600)
  точка 5: (5,1, 5,0500)
  точка 6: (6,0, 6,0400)
  точка 7: (7,0, 7,0300)
  точка 8: (8,0, 8,0200)
  точка 9: (9,0, 9,0100)
  точка 10: (10,0, 10,0000)
функция сериализована в serializable.dat
функция десериализована из serializable.dat
Сравнение исходной и десериализованной функций:
  точка 0: исходная=0,1000, десериализованная=0,
  точка 1: исходная=1,0900, десериализованная=1,
  точка 2: исходная=2,0800, десериализованная=2,
  точка 3: исходная=3,0700, десериализованная=3,
  точка 4: исходная=4,0600, десериализованная=4,
  точка 5: исходная=5,0500, десериализованная=5,
  точка 6: исходная=6,0400, десериализованная=6,
  точка 7: исходная=7,0300, десериализованная=7,
  точка 8: исходная=8,0200, десериализованная=8,
  точка 9: исходная=9,0100, десериализованная=9,
  точка 10: исходная=10,0000, десериализованная=
Сериализация работает правильно - все значения с

Сравнение размеров файлов:
  Сериализация (serializable.dat): 448 байт
  Бинарный формат (test_binary.dat): 84 байт
  Текстовый формат (test_text.txt): 148 байт
```



```
                i, original, deserialized, error);

        if (error > 1e-10) {
            allMatch = false;
        }
    }

    if (allMatch) {
        System.out.println("Сериализация работает правильно - все значения совпадают!");
    } else {
        System.out.println("Ошибка сериализации - значения не совпадают!");
    }

    //анализ файлов
    File serializableFile = new File( pathname: "serializable.dat");
    File binaryFile = new File( pathname: "test_binary.dat");
    File textFile = new File( pathname: "test_text.txt");

    System.out.println("\nСравнение размеров файлов:");
    System.out.printf("  Сериализация (serializable.dat): %d байт%n", serializableFile.length());
    if (binaryFile.exists()) {
        System.out.printf("  Бинарный формат (test_binary.dat): %d байт%n", binaryFile.length());
    }
    if (textFile.exists()) {
        System.out.printf("  Текстовый формат (test_text.txt): %d байт%n", textFile.length());
    }

    System.out.println("\n=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ! ===");

} catch (Exception e) {
    System.out.println("ошибка в новых тестах: " + e.getMessage());
    e.printStackTrace();
}
```

```
  точка 1: (1,1, 1,0900)
  точка 2: (2,1, 2,0800)
  точка 3: (3,1, 3,0700)
  точка 4: (4,1, 4,0600)
  точка 5: (5,1, 5,0500)
  точка 6: (6,0, 6,0400)
  точка 7: (7,0, 7,0300)
  точка 8: (8,0, 8,0200)
  точка 9: (9,0, 9,0100)
  точка 10: (10,0, 10,0000)
функция сериализована в serializable.dat
функция десериализована из serializable.dat
Сравнение исходной и десериализованной функций:
  точка 0: исходная=0,1000, десериализованная=0,
  точка 1: исходная=1,0900, десериализованная=1,
  точка 2: исходная=2,0800, десериализованная=2,
  точка 3: исходная=3,0700, десериализованная=3,
  точка 4: исходная=4,0600, десериализованная=4,
  точка 5: исходная=5,0500, десериализованная=5,
  точка 6: исходная=6,0400, десериализованная=6,
  точка 7: исходная=7,0300, десериализованная=7,
  точка 8: исходная=8,0200, десериализованная=8,
  точка 9: исходная=9,0100, десериализованная=9,
  точка 10: исходная=10,0000, десериализованная=
Сериализация работает правильно - все значения с

Сравнение размеров файлов:
  Сериализация (serializable.dat): 448 байт
  Бинарный формат (test_binary.dat): 84 байт
  Текстовый формат (test_text.txt): 148 байт

=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ! ===

Process finished with exit code 0
```