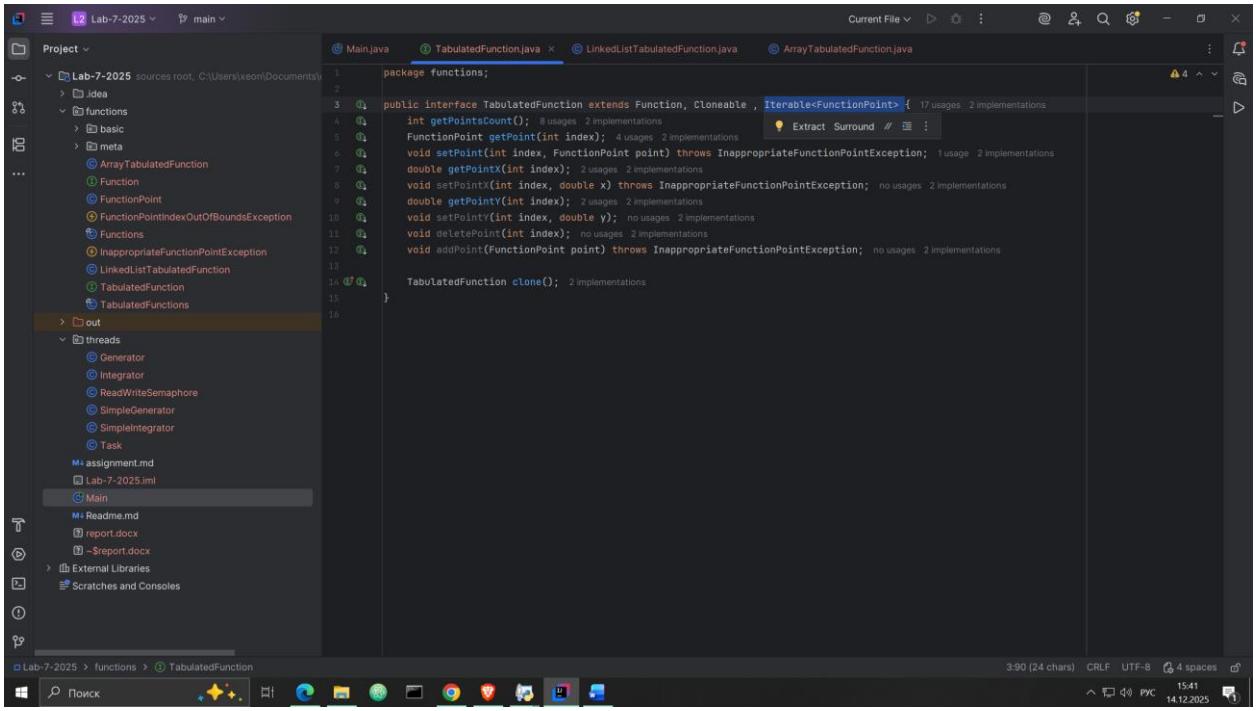
Отчёт по лабораторной работе №7

Тенигин Валерий 6204-010302D

Задание 1
Добавил в интерфейс TabulatedFunction наследование от Iterable<FunctionPoint>

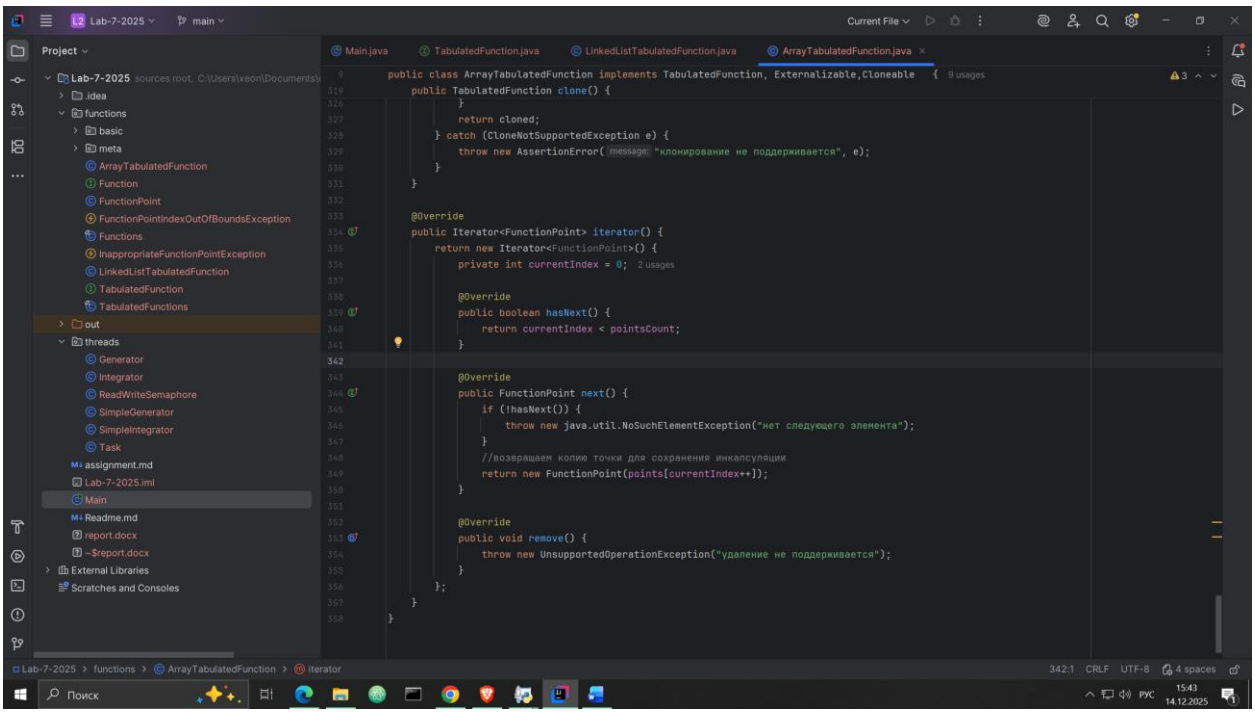


Добавил методы:

iterator() с анонимным классом

next() возвращает копии точек для сохранения инкапсуляции и выбрасывает NoSuchElementException при отсутствии элементов

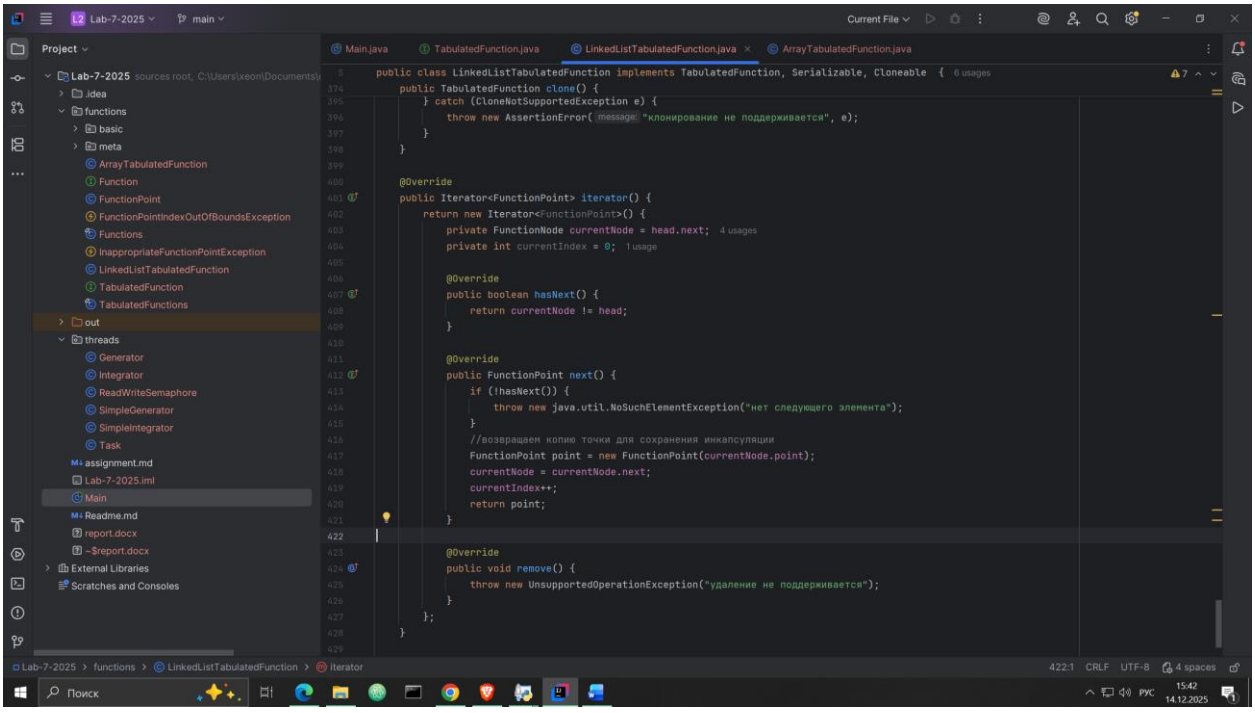
remove() выбрасывает UnsupportedOperationException



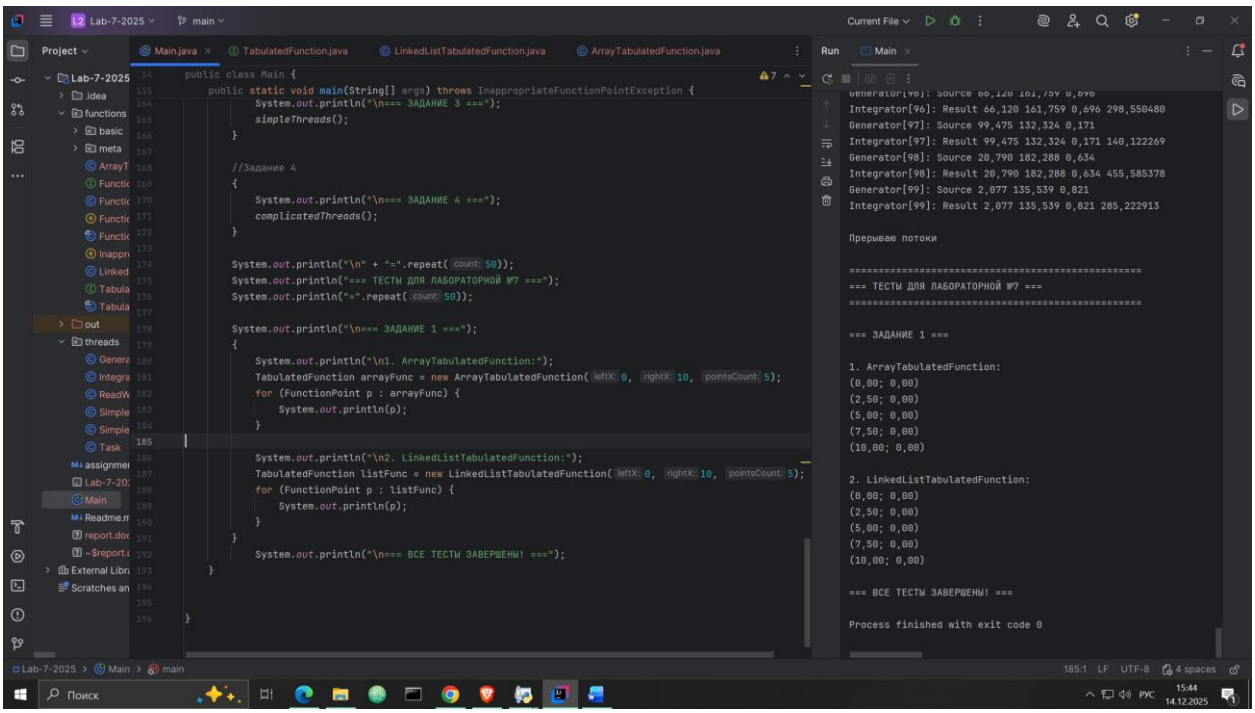
Добавил метод iterator() с анонимным классом

next() возвращает копии точек для сохранения инкапсуляции и выбрасывает NoSuchElementException при отсутствии элементов

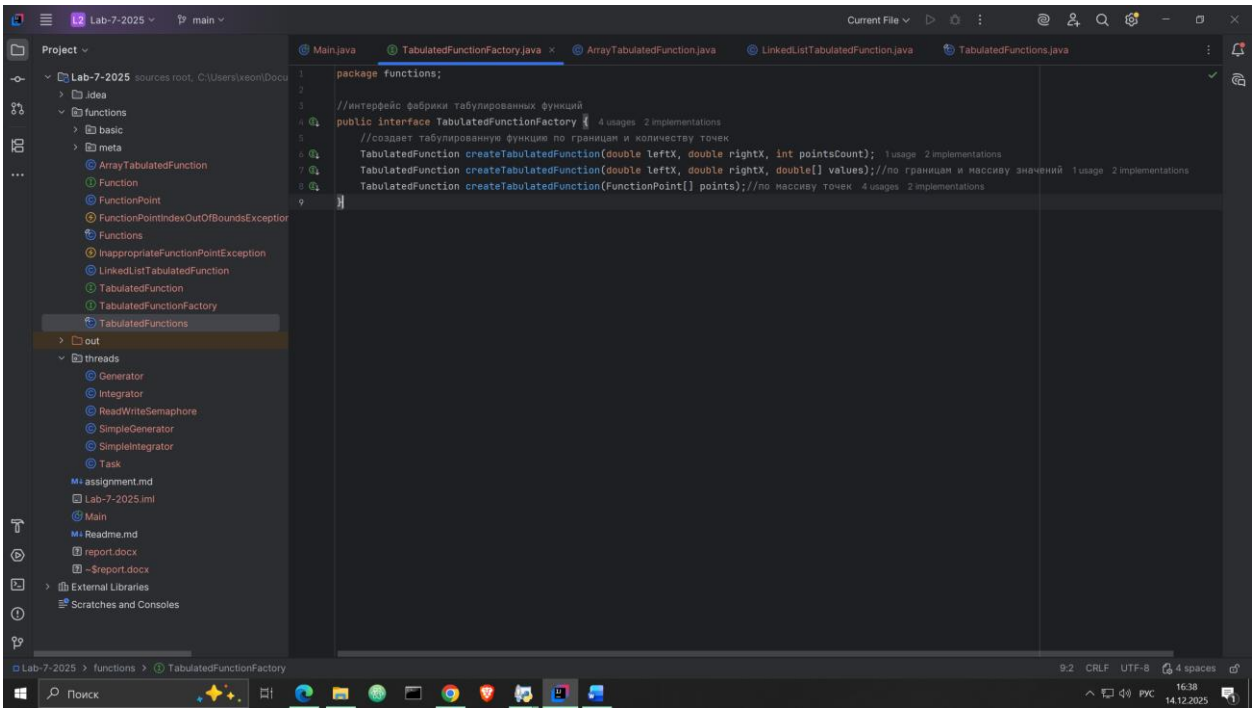
remove() выбрасывает UnsupportedOperationException



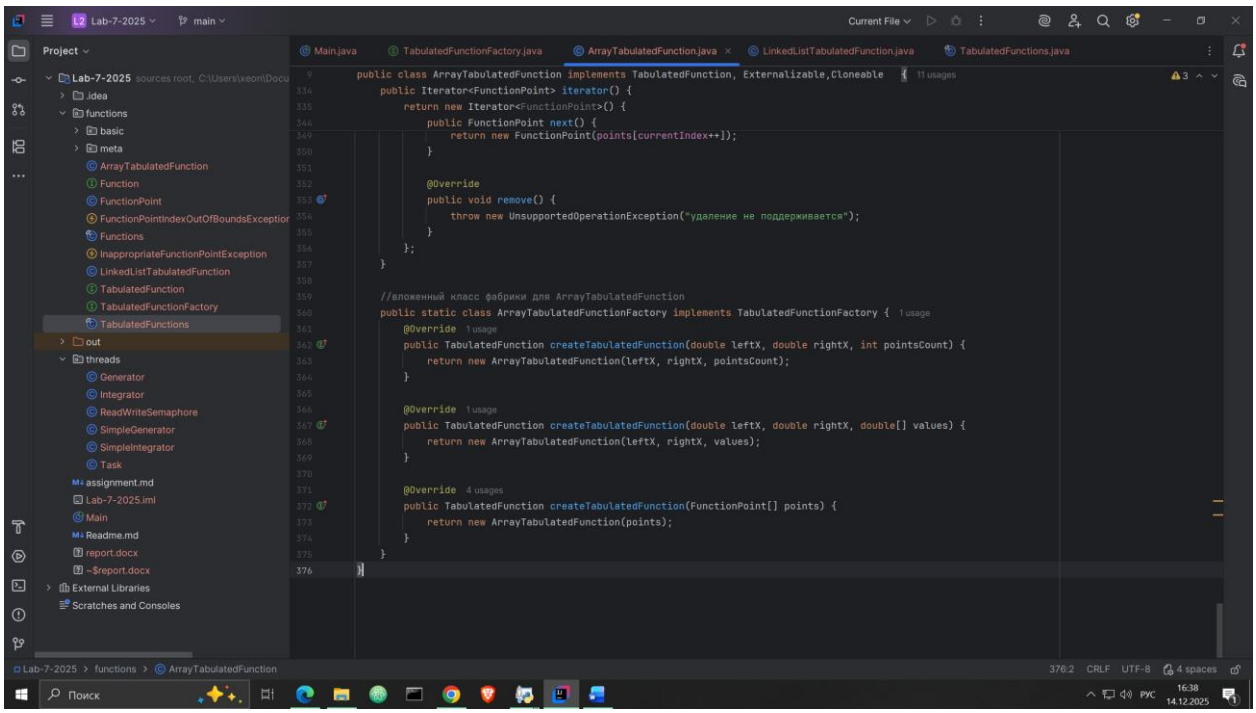
Тестирование

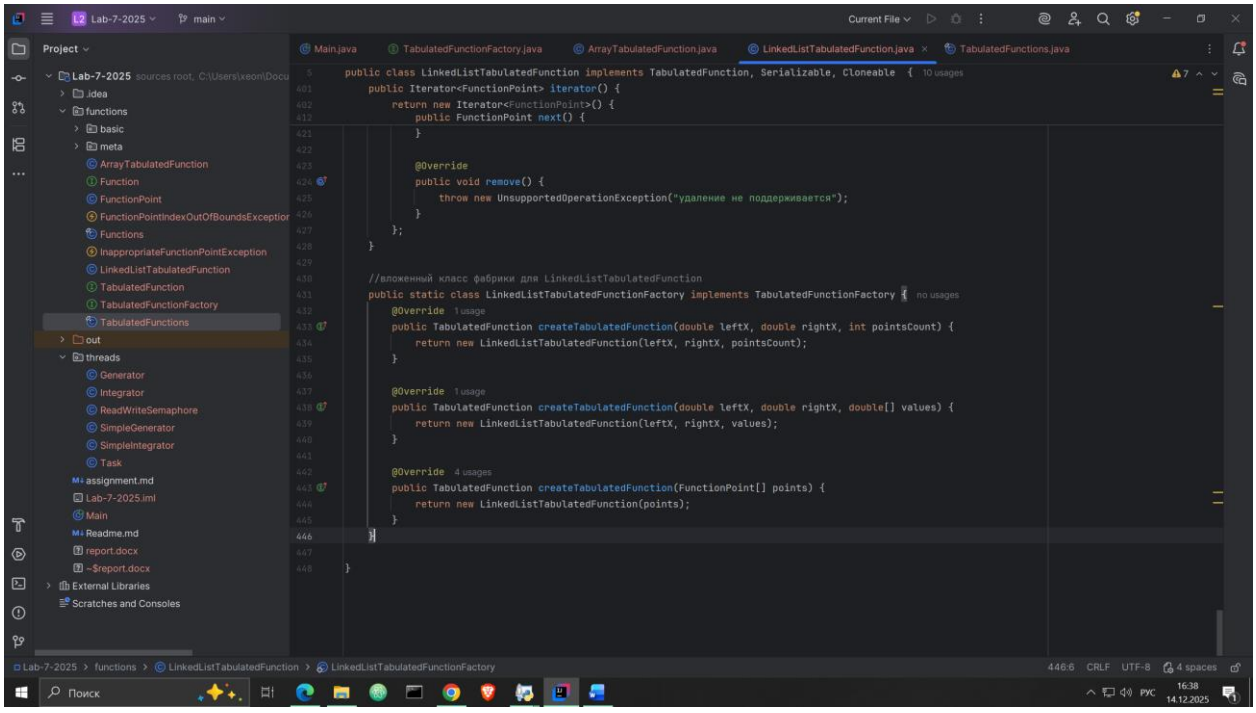


Задание 2
Создал интерфейс TabulatedFunctionFactory в пакете functions, который содержит три перегруженных метода createTabulatedFunction() где параметры методов соответствуют конструкторам табулированных функций



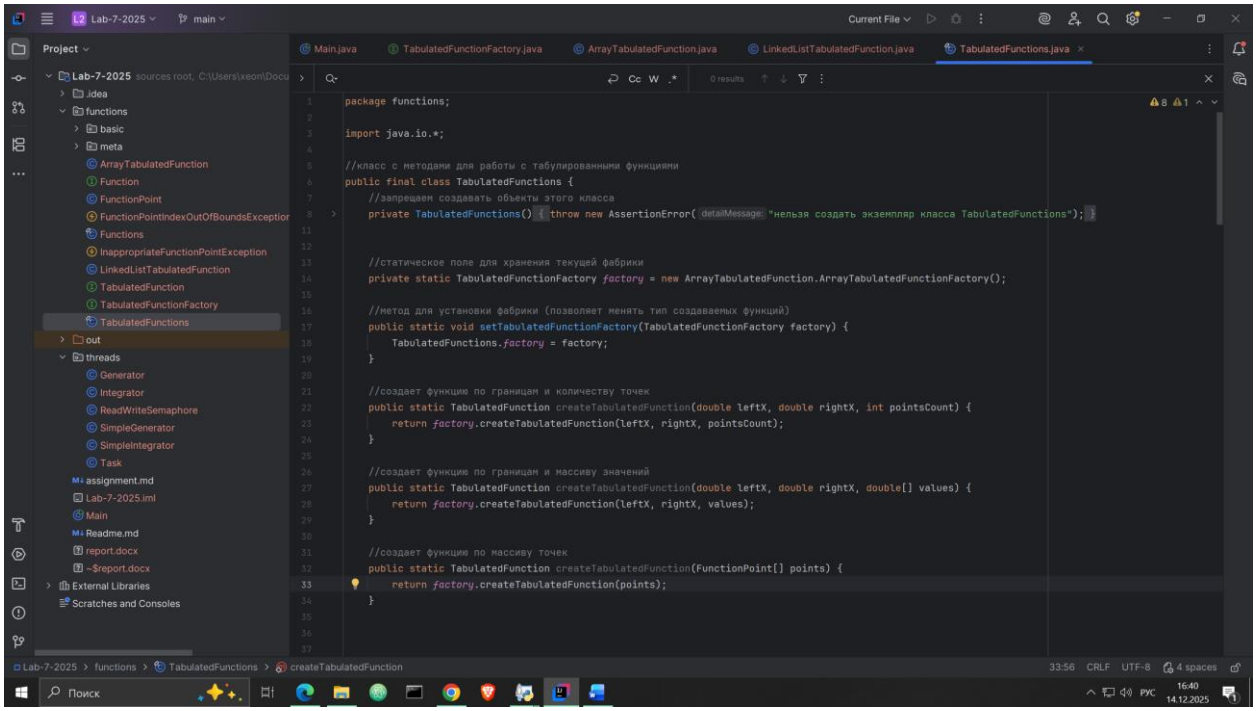
Создал два класса которые реализуют интерфейс TabulatedFunctionFactory





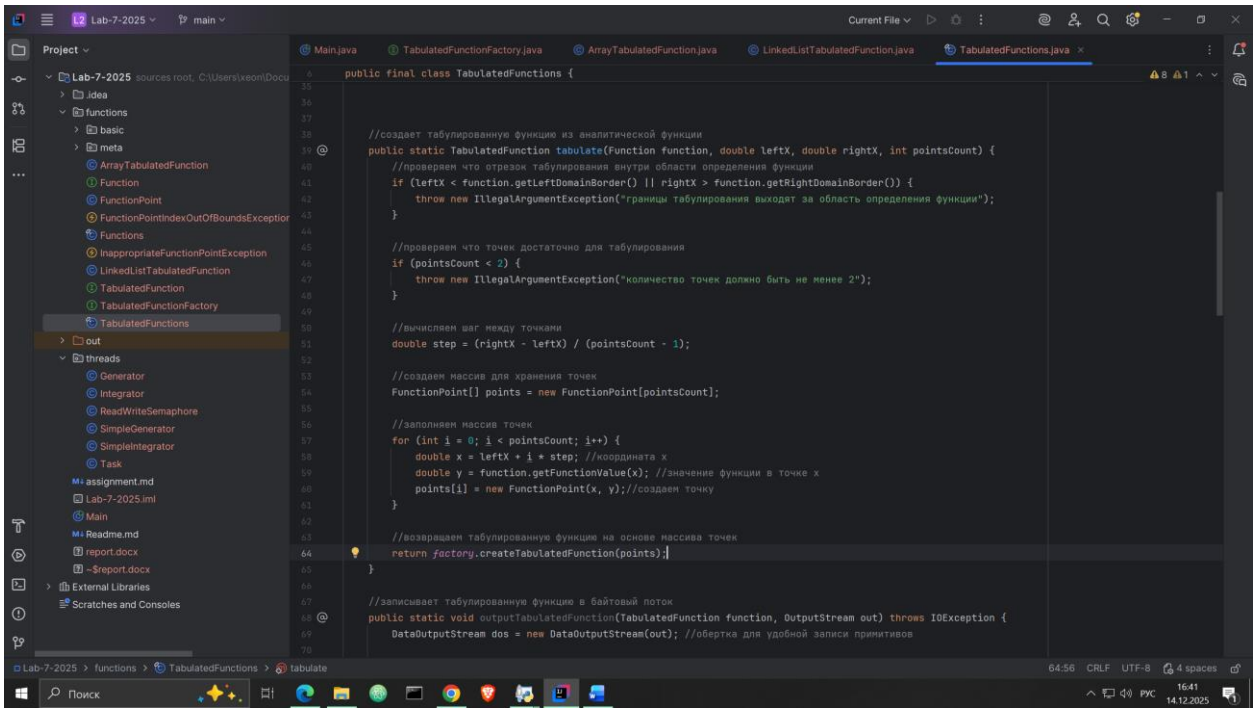
Добавил
поле factory для хранения текущей фабрики создания функций

метод setTabulatedFunctionFactory() для смены фабрики
три фабричных метода createTabulatedFunction()

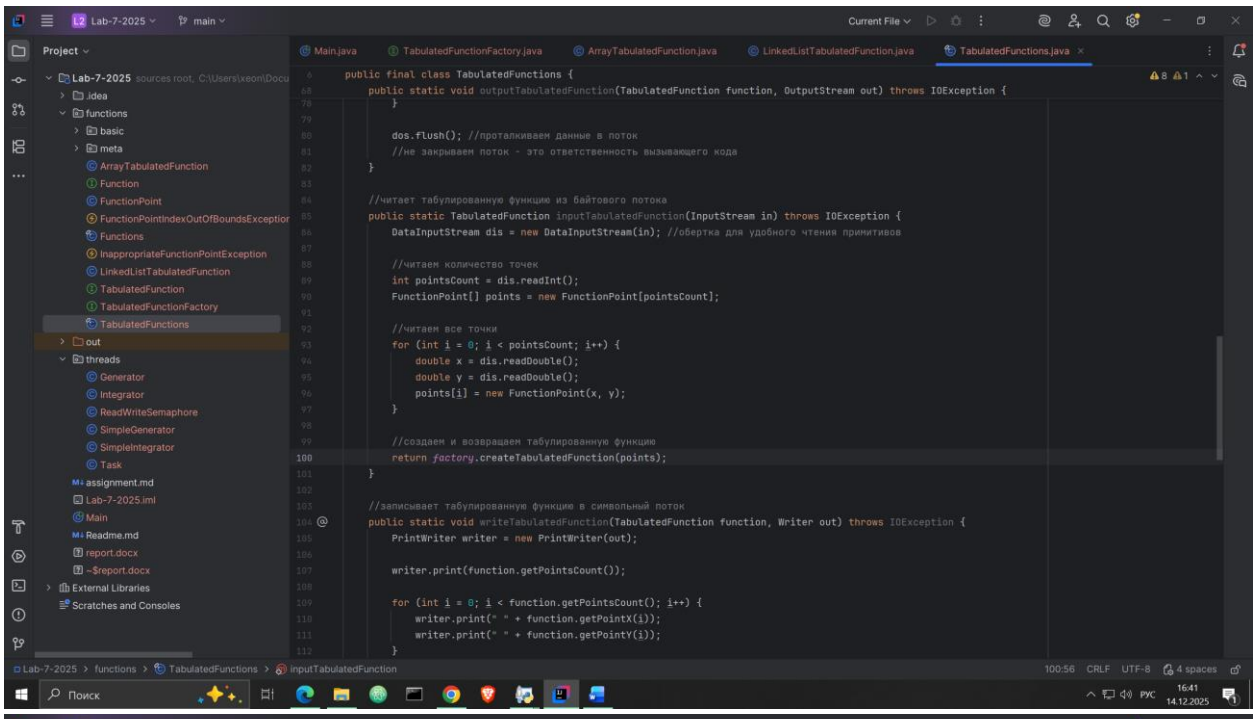


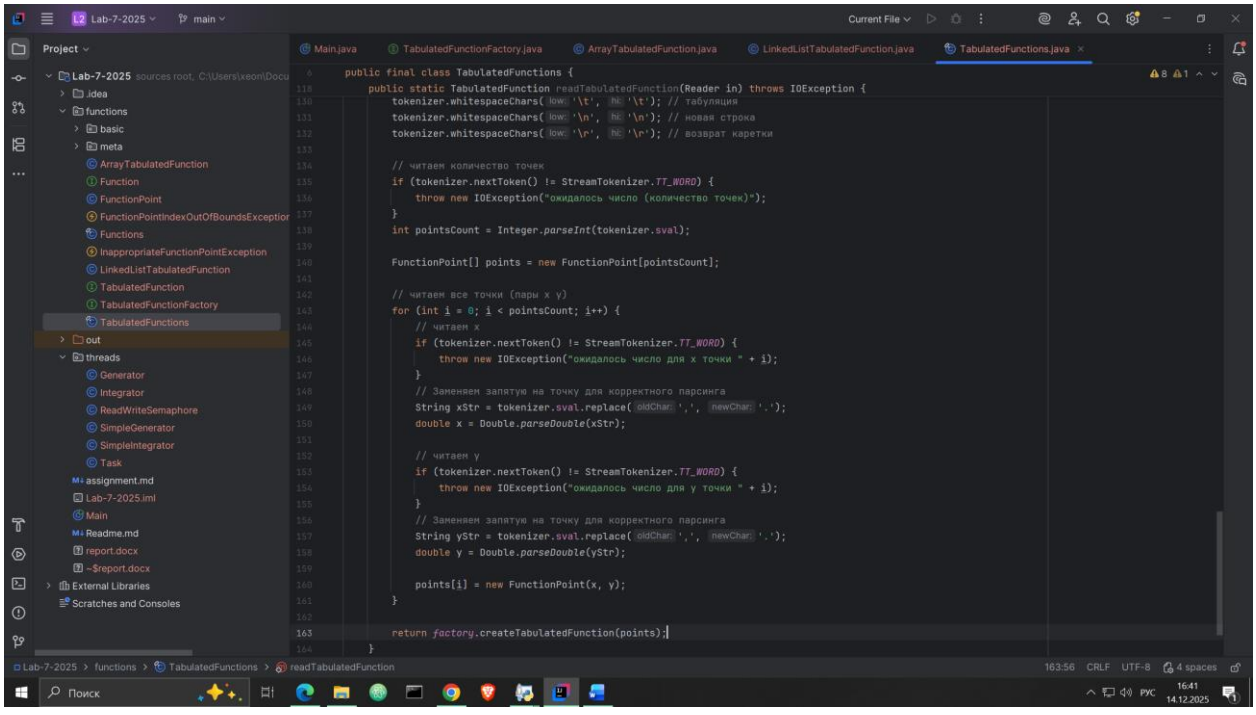
```java
package functions;

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( detailMessage: "нельзя создать экземпляр класса TabulatedFunctions"); }


    //статическое поле для хранения текущей фабрики
    private static TabulatedFunctionFactory factory = new ArrayTabulatedFunction.ArrayTabulatedFunctionFactory();

    //метод для установки фабрики (позволяет менять тип создаваемых функций)
    public static void setTabulatedFunctionFactory(TabulatedFunctionFactory factory) {
        TabulatedFunctions.factory = factory;
    }

    //создает функцию по границам и количеству точек
    public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) {
        return factory.createTabulatedFunction(leftX, rightX, pointsCount);
    }

    //создает функцию по границам и массиву значений
    public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) {
        return factory.createTabulatedFunction(leftX, rightX, values);
    }

    //создает функцию по массиву точек
    public static TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {
        return factory.createTabulatedFunction(points);
    }
```

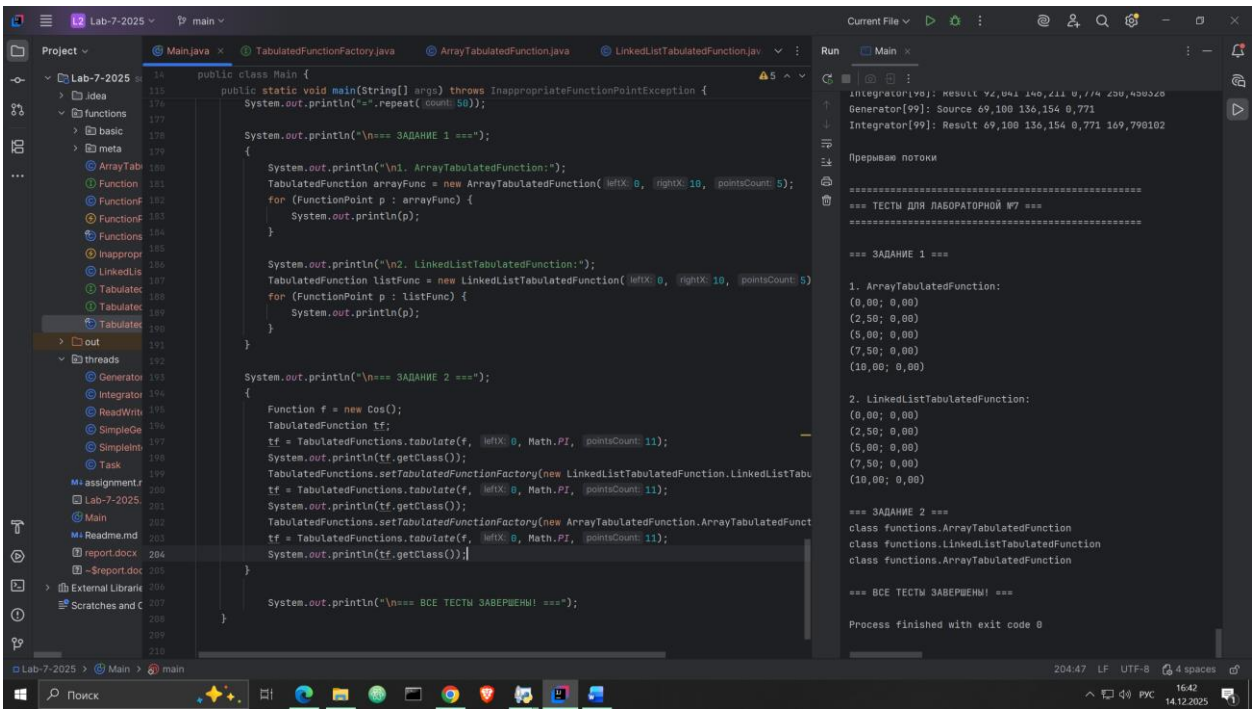
Заменил все new ArrayTabulatedFunction() в методах tabulate(), inputTabulatedFunction() и readTabulatedFunction() на вызовы factory.createTabulatedFunction()



```java
public final class 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
        return factory.createTabulatedFunction(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
```


Lab-7-2025
main
Current File
Project
Main.java
TabulatedFunctionFactory.java
ArrayTabulatedFunction.java
LinkedListTabulatedFunction.java
TabulatedFunctions.java
Lab-7-2025 sources root, C:\Users\xeom\Docu
.idea
functions
basic
meta
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBoundsException
Functions
InappropriateFunctionPointException
LinkedListTabulatedFunction
TabulatedFunction
TabulatedFunctionFactory
TabulatedFunctions
out
threads
Generator
Integrator
ReadWriteSemaphore
SimpleGenerator
SimpleIntegrator
Task
assignment.md
Lab-7-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
public final class TabulatedFunctions {
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException {
}
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
return factory.createTabulatedFunction(points);
}
//записывает табулированную функцию в символьный поток
public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException {
PrintWriter writer = new PrintWriter(out);
writer.print(function.getPointsCount());
for (int i = 0; i < function.getPointsCount(); i++) {
writer.print(" " + function.getPointX(i));
writer.print(" " + function.getPointY(i));
}
Lab-7-2025 > functions > TabulatedFunctions > inputTabulatedFunction
100:56 CRLF UTF-8 4 spaces
Поиск
РУС
16:41
14.12.2025


Lab-7-2025
main
Current File
Project
Main.java
TabulatedFunctionFactory.java
ArrayTabulatedFunction.java
LinkedListTabulatedFunction.java
TabulatedFunctions.java
Lab-7-2025 sources root, C:\Users\xeom\Docu
.idea
functions
basic
meta
ArrayTabulatedFunction
Function
FunctionPoint
FunctionPointIndexOutOfBoundsException
Functions
InappropriateFunctionPointException
LinkedListTabulatedFunction
TabulatedFunction
TabulatedFunctionFactory
TabulatedFunctions
out
threads
Generator
Integrator
ReadWriteSemaphore
SimpleGenerator
SimpleIntegrator
Task
assignment.md
Lab-7-2025.iml
Main
Readme.md
report.docx
~$report.docx
External Libraries
Scratches and Consoles
public final class TabulatedFunctions {
public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException {
tokenizer.whitespaceChars(low: '\t', hi: '\t'); // табуляция
tokenizer.whitespaceChars(low: '\n', hi: '\n'); // новая строка
tokenizer.whitespaceChars(low: '\r', hi: '\r'); // возврат каретки
// читаем количество точек
if (tokenizer.nextToken() != StreamTokenizer.TT_WORD) {
throw new IOException("ожидалось число (количество точек)");
}
int pointsCount = Integer.parseInt(tokenizer.sval);
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_WORD) {
throw new IOException("ожидалось число для x точки " + i);
}
// Заменяем запятую на точку для корректного парсинга
String xStr = tokenizer.sval.replace(oldChar: ',', newChar: '.');
double x = Double.parseDouble(xStr);
// читаем y
if (tokenizer.nextToken() != StreamTokenizer.TT_WORD) {
throw new IOException("ожидалось число для y точки " + i);
}
// Заменяем запятую на точку для корректного парсинга
String yStr = tokenizer.sval.replace(oldChar: ',', newChar: '.');
double y = Double.parseDouble(yStr);
points[i] = new FunctionPoint(x, y);
}
return factory.createTabulatedFunction(points);
}
Lab-7-2025 > functions > TabulatedFunctions > readTabulatedFunction
163:56 CRLF UTF-8 4 spaces
Поиск
РУС
16:41
14.12.2025

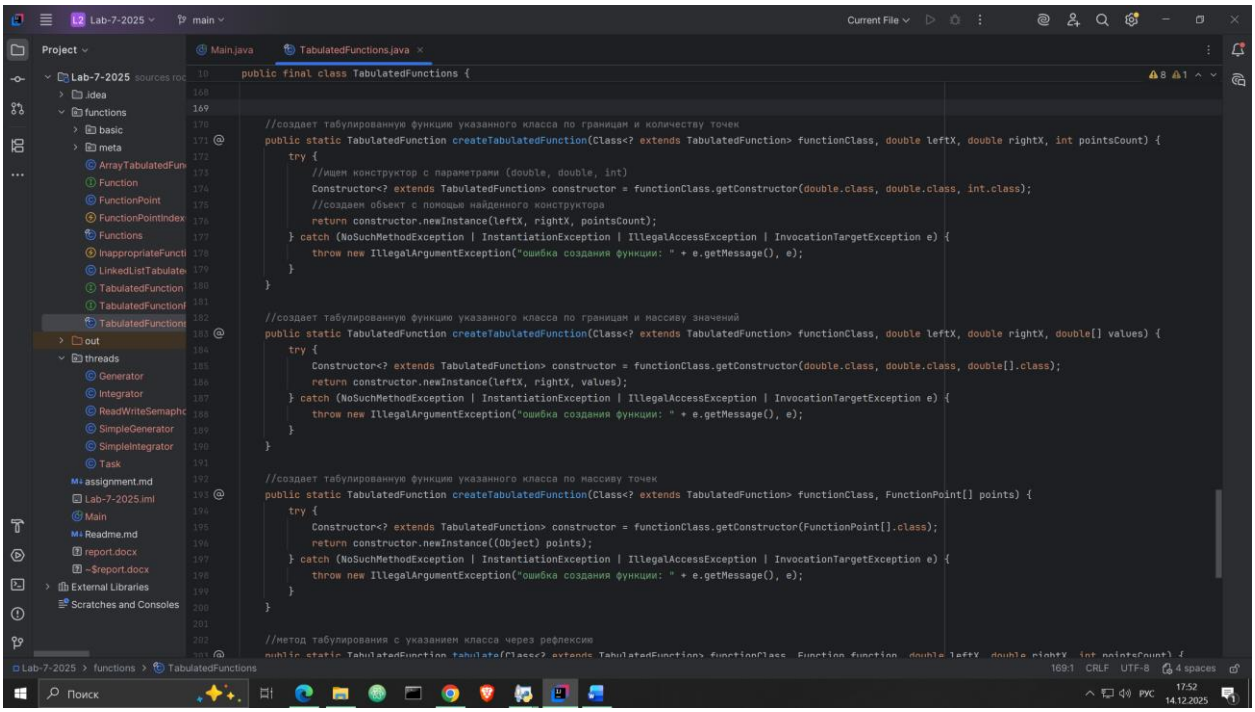


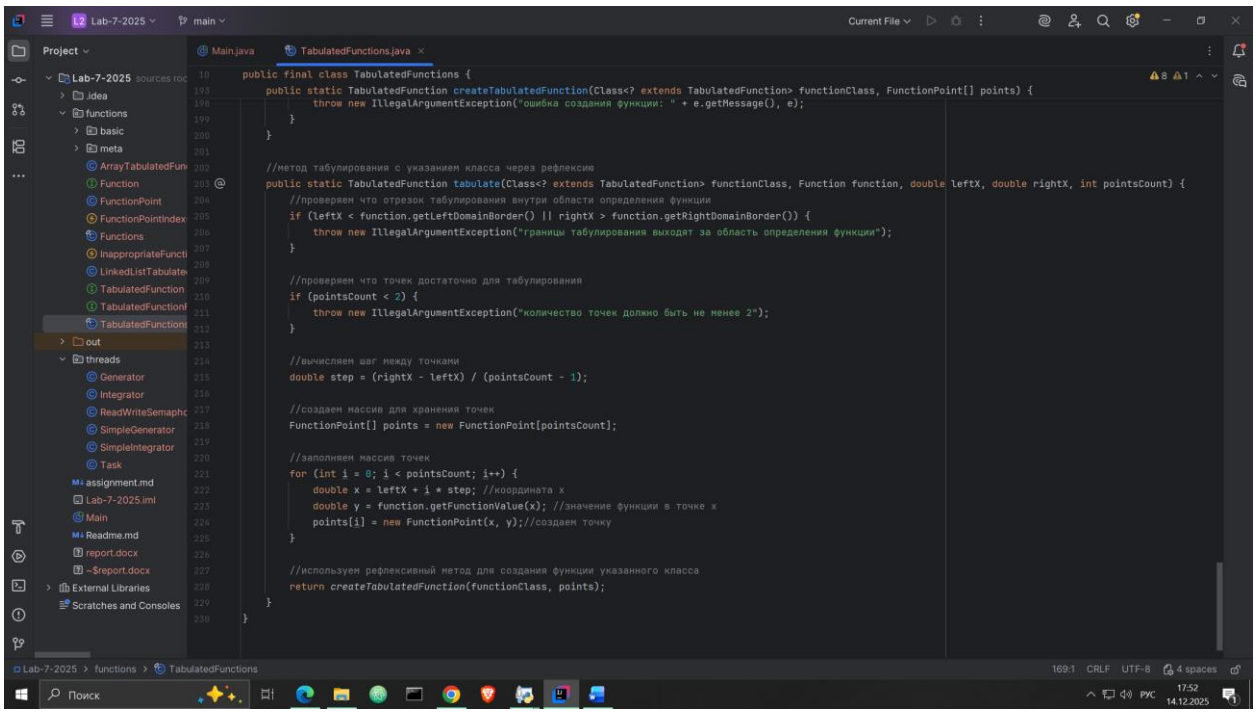
Задание 3

Добавил три перегруженных метода createTabulatedFunction() для рефлексивного создания

Добавил метод tabulate() с параметром Class для рефлексивного табулирования

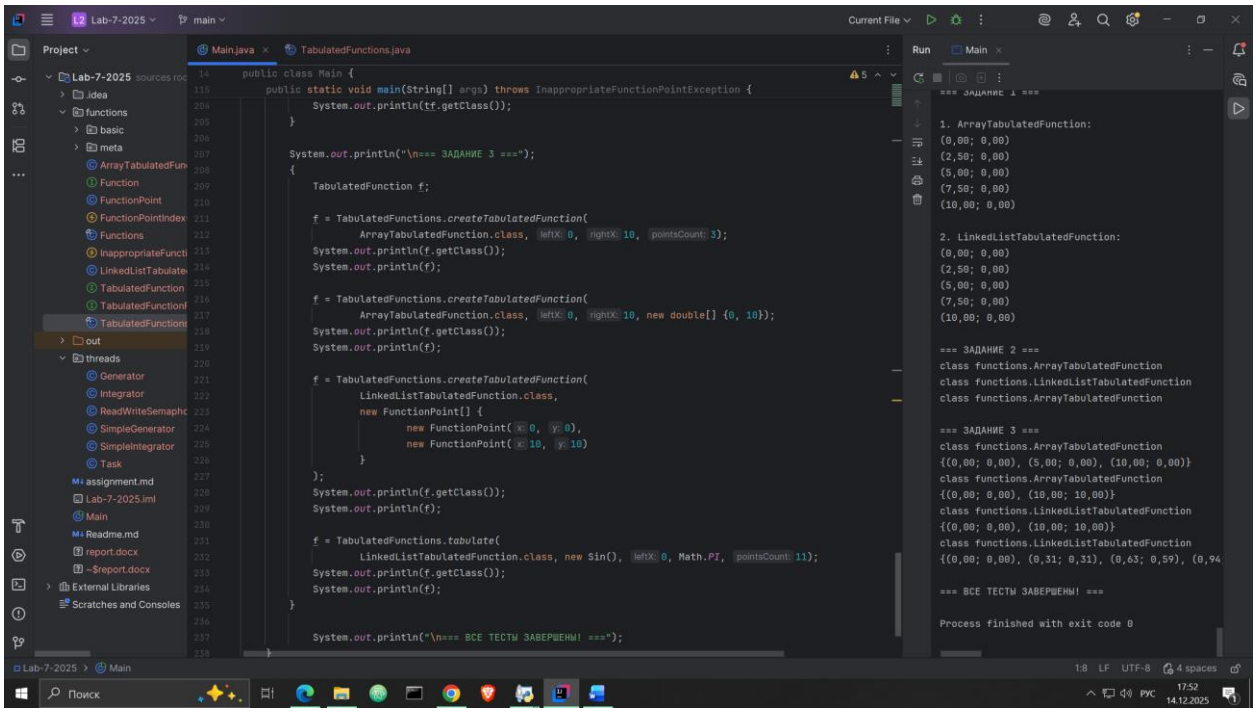
Добавил обработку исключений рефлексии с преобразованием в IllegalArgumentException





```java
public final class TabulatedFunctions {
    public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> functionClass, FunctionPoint[] points) {
            throw new IllegalArgumentException("ошибка создания функции: " + e.getMessage(), e);
        }
    }

    //метод табулирования с указанием класса через рефлексию
    public static TabulatedFunction tabulate(Class<? extends TabulatedFunction> functionClass, 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
        }

        //используем рефлексивный метод для создания функции указанного класса
        return createTabulatedFunction(functionClass, points);
    }
}
```

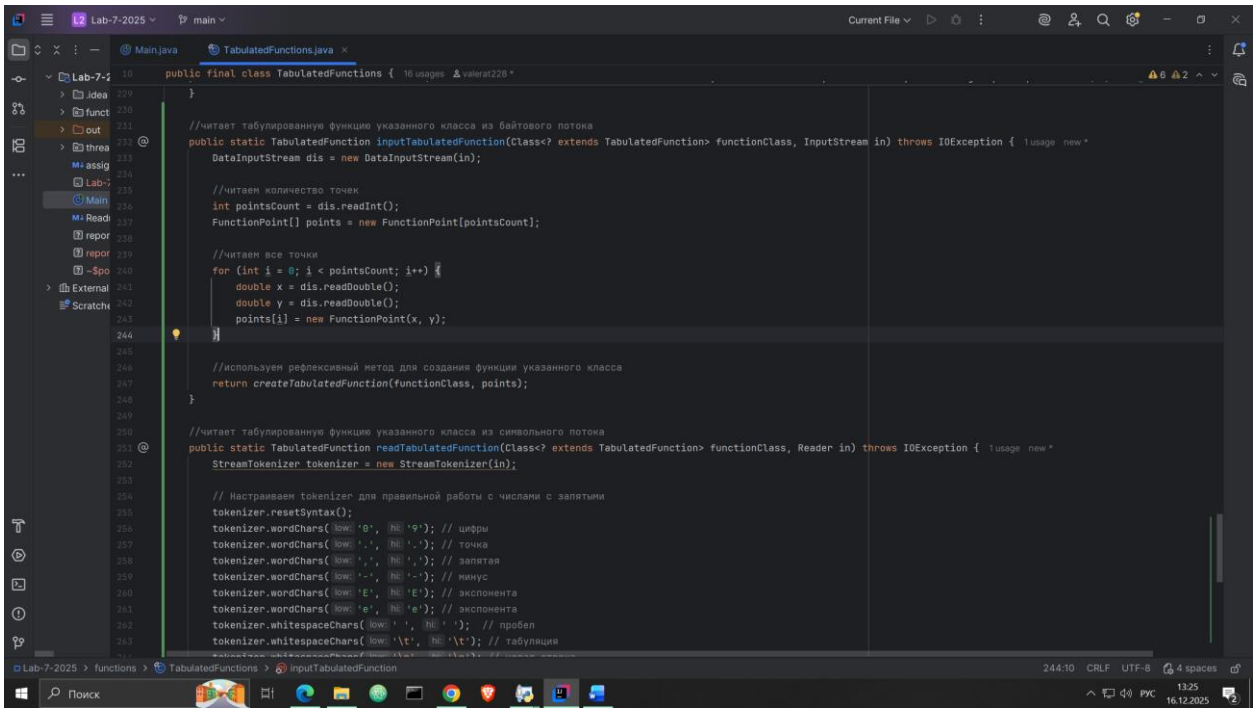
Тестироование



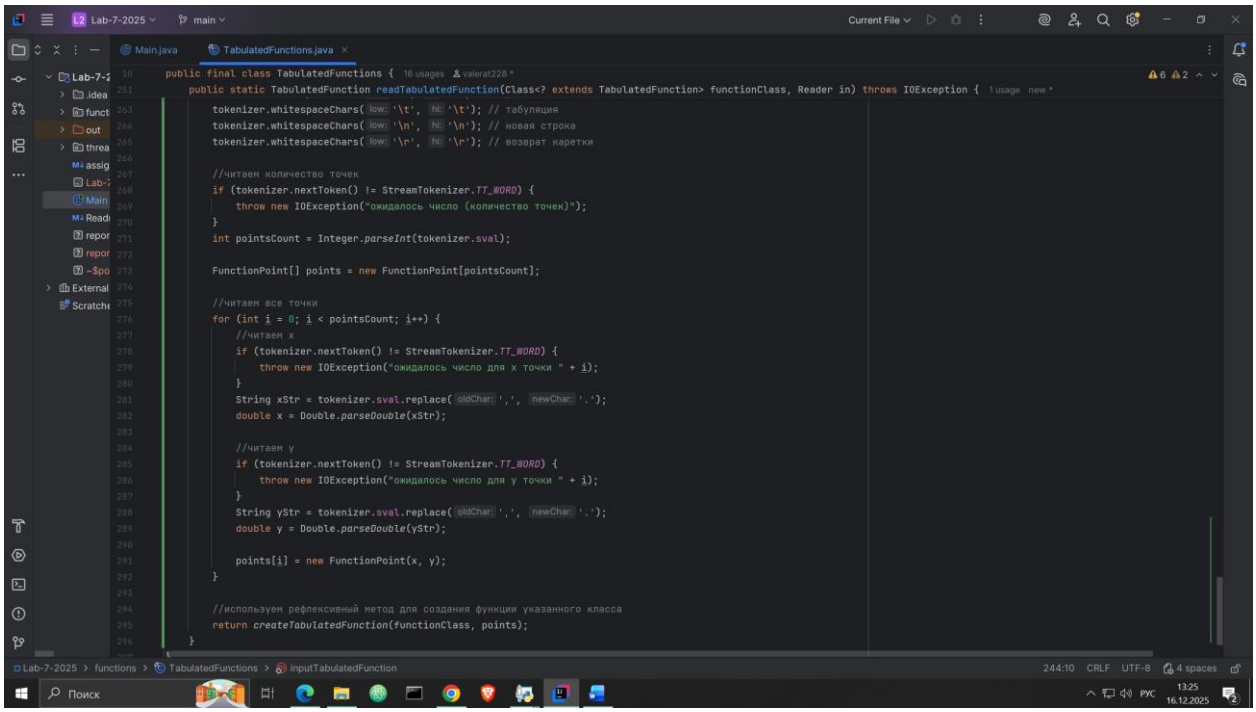
```java
public class Main {
    public static void main(String[] args) throws InappropriateFunctionPointException {
            System.out.println(tf.getClass());
        }

        System.out.println("\n=== ЗАДАНИЕ 3 ===");
        {
            TabulatedFunction f;

            f = TabulatedFunctions.createTabulatedFunction(
                    ArrayTabulatedFunction.class, leftX: 0, rightX: 10, pointsCount: 3);
            System.out.println(f.getClass());
            System.out.println(f);

            f = TabulatedFunctions.createTabulatedFunction(
                    ArrayTabulatedFunction.class, leftX: 0, rightX: 10, new double[] {0, 10});
            System.out.println(f.getClass());
            System.out.println(f);

            f = TabulatedFunctions.createTabulatedFunction(
                    LinkedListTabulatedFunction.class,
                    new FunctionPoint[] {
                            new FunctionPoint( x: 0, y: 0),
                            new FunctionPoint( x: 10, y: 10)
                    }
            );
            System.out.println(f.getClass());
            System.out.println(f);

            f = TabulatedFunctions.tabulate(
                    LinkedListTabulatedFunction.class, new Sin(), leftX: 0, Math.PI, pointsCount: 11);
            System.out.println(f.getClass());
            System.out.println(f);
        }

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

```
=== ЗАДАНИЕ 1 ===

1. ArrayTabulatedFunction:
{0,00; 0,00}
{2,50; 0,00}
{5,00; 0,00}
{7,50; 0,00}
{10,00; 0,00}

2. LinkedListTabulatedFunction:
{0,00; 0,00}
{2,50; 0,00}
{5,00; 0,00}
{7,50; 0,00}
{10,00; 0,00}

=== ЗАДАНИЕ 2 ===
class functions.ArrayTabulatedFunction
class functions.LinkedListTabulatedFunction
class functions.ArrayTabulatedFunction

=== ЗАДАНИЕ 3 ===
class functions.ArrayTabulatedFunction
{(0,00; 0,00), (5,00; 0,00), (10,00; 0,00)}
class functions.ArrayTabulatedFunction
{(0,00; 0,00), (10,00; 10,00)}
class functions.LinkedListTabulatedFunction
{(0,00; 0,00), (10,00; 10,00)}
class functions.LinkedListTabulatedFunction
{(0,00; 0,00), (0,31; 0,31), (0,63; 0,59), (0,94

=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ! ===

Process finished with exit code 0
```

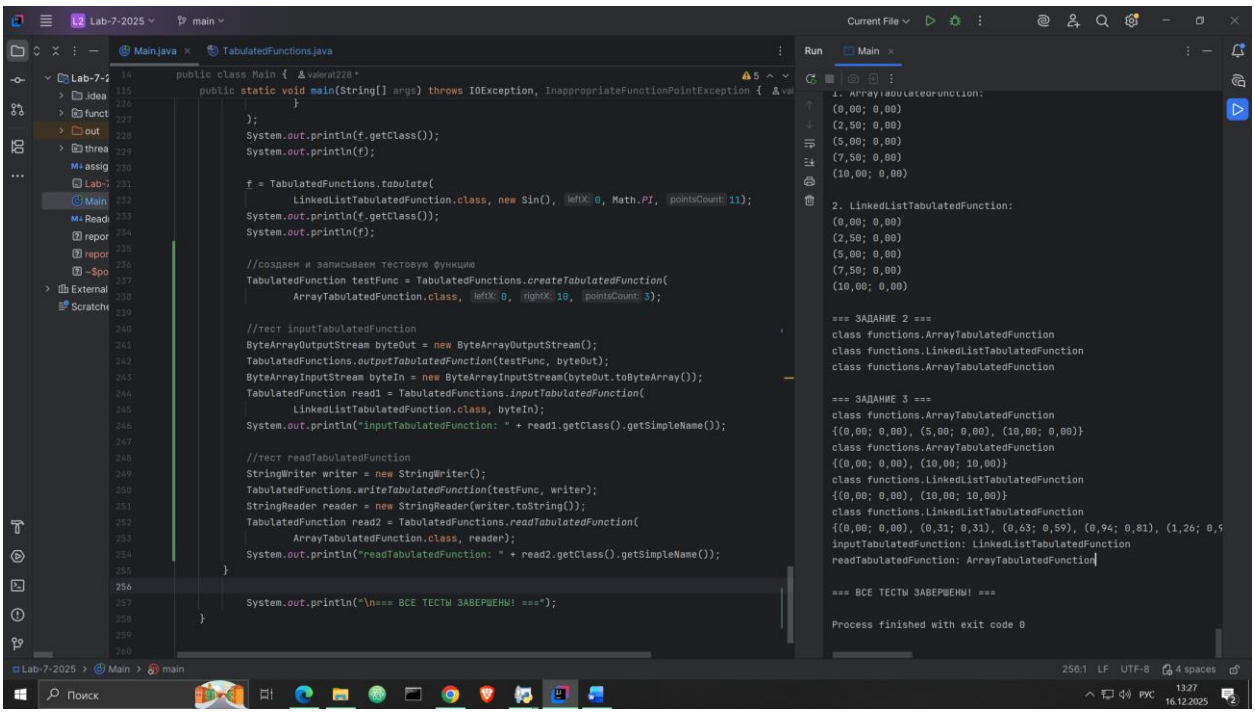
Дополнение



```
public final class TabulatedFunctions {  16 usages  valerat228 *
    }

    //читает табулированную функцию указанного класса из байтового потока
    public static TabulatedFunction inputTabulatedFunction(Class<? extends TabulatedFunction> functionClass, InputStream in) throws IOException {  1 usage  new *
        DataInputStream dis = new DataInputStream(in);

        //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);
        }

        //используем рефлексивный метод для создания функции указанного класса
        return createTabulatedFunction(functionClass, points);
    }

    //читает табулированную функцию указанного класса из символьного потока
    public static TabulatedFunction readTabulatedFunction(Class<? extends TabulatedFunction> functionClass, Reader in) throws IOException {  1 usage  new *
        StreamTokenizer tokenizer = new StreamTokenizer(in);

        // Настраиваем tokenizer для правильной работы с числами с запятыми
        tokenizer.resetSyntax();
        tokenizer.wordChars( low: '0',  hi: '9'); // цифры
        tokenizer.wordChars( low: '.',  hi: '.'); // точка
        tokenizer.wordChars( low: ',',  hi: ','); // запятая
        tokenizer.wordChars( low: '-',  hi: '-'); // минус
        tokenizer.wordChars( low: 'E',  hi: 'E'); // экспонента
        tokenizer.wordChars( low: 'e',  hi: 'e'); // экспонента
        tokenizer.whitespaceChars( low: ' ',  hi: ' ');  // пробел
        tokenizer.whitespaceChars( low: '\t',  hi: '\t'); // табуляция
```



```
public final class TabulatedFunctions {  16 usages  valerat228 *
    public static TabulatedFunction readTabulatedFunction(Class<? extends TabulatedFunction> functionClass, Reader in) throws IOException {  1 usage  new *
        tokenizer.whitespaceChars( low: '\t',  hi: '\t'); // табуляция
        tokenizer.whitespaceChars( low: '\n',  hi: '\n'); // новая строка
        tokenizer.whitespaceChars( low: '\r',  hi: '\r'); // возврат каретки

        //читаем количество точек
        if (tokenizer.nextToken() != StreamTokenizer.TT_WORD) {
            throw new IOException("ожидалось число (количество точек)");
        }
        int pointsCount = Integer.parseInt(tokenizer.sval);

        FunctionPoint[] points = new FunctionPoint[pointsCount];

        //читаем все точки
        for (int i = 0; i < pointsCount; i++) {
            //читаем x
            if (tokenizer.nextToken() != StreamTokenizer.TT_WORD) {
                throw new IOException("ожидалось число для x точки " + i);
            }
            String xStr = tokenizer.sval.replace( oldChar: ',',  newChar: '.');
            double x = Double.parseDouble(xStr);

            //читаем y
            if (tokenizer.nextToken() != StreamTokenizer.TT_WORD) {
                throw new IOException("ожидалось число для y точки " + i);
            }
            String yStr = tokenizer.sval.replace( oldChar: ',',  newChar: '.');
            double y = Double.parseDouble(yStr);

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

        //используем рефлексивный метод для создания функции указанного класса
        return createTabulatedFunction(functionClass, points);
    }
```


Lab-7-2025
main
Current File
Main.java
TabulatedFunctions.java
Run
Main
public class Main { valerat228 *
public static void main(String[] args) throws IOException, InappropriateFunctionPointException {
);
System.out.println(f.getClass());
System.out.println(f);
f = TabulatedFunctions.tabulate(
LinkedListTabulatedFunction.class, new Sin(), leftX: 0, Math.PI, pointsCount: 11);
System.out.println(f.getClass());
System.out.println(f);
//создаем и записываем тестовую функцию
TabulatedFunction testFunc = TabulatedFunctions.createTabulatedFunction(
ArrayTabulatedFunction.class, leftX: 0, rightX: 10, pointsCount: 3);
//тест inputTabulatedFunction
ByteArrayOutputStream byteOut = new ByteArrayOutputStream();
TabulatedFunctions.outputTabulatedFunction(testFunc, byteOut);
ByteArrayInputStream byteIn = new ByteArrayInputStream(byteOut.toByteArray());
TabulatedFunction read1 = TabulatedFunctions.inputTabulatedFunction(
LinkedListTabulatedFunction.class, byteIn);
System.out.println("inputTabulatedFunction: " + read1.getClass().getSimpleName());
//тест readTabulatedFunction
StringWriter writer = new StringWriter();
TabulatedFunctions.writeTabulatedFunction(testFunc, writer);
StringReader reader = new StringReader(writer.toString());
TabulatedFunction read2 = TabulatedFunctions.readTabulatedFunction(
ArrayTabulatedFunction.class, reader);
System.out.println("readTabulatedFunction: " + read2.getClass().getSimpleName());
System.out.println("\n=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ! ===");
(0,00; 0,00)
(2,50; 0,00)
(5,00; 0,00)
(7,50; 0,00)
(10,00; 0,00)
2. LinkedListTabulatedFunction:
(0,00; 0,00)
(2,50; 0,00)
(5,00; 0,00)
(7,50; 0,00)
(10,00; 0,00)
=== ЗАДАНИЕ 2 ===
class functions.ArrayTabulatedFunction
class functions.LinkedListTabulatedFunction
class functions.ArrayTabulatedFunction
=== ЗАДАНИЕ 3 ===
class functions.ArrayTabulatedFunction
{(0,00; 0,00), (5,00; 0,00), (10,00; 0,00)}
class functions.ArrayTabulatedFunction
{(0,00; 0,00), (10,00; 10,00)}
class functions.LinkedListTabulatedFunction
{(0,00; 0,00), (10,00; 10,00)}
class functions.LinkedListTabulatedFunction
{(0,00; 0,00), (0,31; 0,31), (0,63; 0,59), (0,94; 0,81), (1,26; 0,
inputTabulatedFunction: LinkedListTabulatedFunction
readTabulatedFunction: ArrayTabulatedFunction
=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ! ===
Process finished with exit code 0
Lab-7-2025 > Main > main
256:1 LF UTF-8 4 spaces
Поиск
РУС
13:27
16.12.2025