

Министерство науки и высшего образования
Российской

Федерации федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский
университет имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе № 4

Дисциплина: «ООП»

Тема «Наследование. Ввод и вывод»

Выполнил: Сучков Борис Антонович

Группа: 6201-120303D

Самара 2025

Задание на лабораторную работу

Расширить возможности пакета для работы с функциями одной переменной добавив интерфейсы и классы для аналитически заданных функций, а также методы ввода и вывода табулированных функций.

Не забывайте использовать машинный эпсилон при сравнении вещественных чисел.

Задания

Задание 1

В классах `ArrayTabulatedFunction` И `LinkedListTabulatedFunction` добавьте конструкторы, получающие сразу все точки функции в виде массива объектов типа `FunctionPoint`. Если точек задано меньше двух, или если точки в массиве не упорядочены по значению абсциссы, конструкторы должны выбрасывать исключение `IllegalArgumentException`. При написании конструкторов обеспечьте корректную инкапсуляцию.

Задание 2

В пакете `functions` создайте интерфейс `Function`, описывающий функции одной переменной и содержащий следующие методы:

- `public double getLeftDomainBorder()` – возвращает значение левой границы области определения функции;
- `public double getRightDomainBorder()` – возвращает значение правой границы области определения функции;
- `public double getFunctionValue(double x)` – возвращает значение функции в заданной точке.

Исключите соответствующие методы из интерфейса `TabulatedFunction` и сделайте так, чтобы он расширял интерфейс `Function`. Теперь табулированные функции будут частным случаем функций одной переменной.

Задание 3

Создайте пакет `functions.basic`, в нём будут описаны классы ряда функций, заданных аналитически.

Создайте в пакете публичный класс `Exp`, объекты которого должны вычислять значение экспоненты. Класс должен реализовывать интерфейс `Function`. Для вычисления экспоненты следует воспользоваться методом `Math.exp()`, а для возвращения значений границ области определения – константами из класса `Double`.

Аналогично, создайте класс `Log`, объекты которого должны вычислять значение логарифма по заданному основанию. Основание должно передаваться как параметр конструктора. Для вычисления логарифма следует воспользоваться методом `Math.log()`.

Прежде, чем перейти к описанию классов для тригонометрических функций (синуса, косинуса и тангенса), обратите внимание на то, что область определения этих функций совпадает, поэтому описывать одинаковые методы в этих классах будет достаточно странным. Проще будет описать базовый класс с реализацией этих методов, а классы конкретных функций наследовать от него.

Создайте класс `TrigonometricFunction`, реализующий интерфейс `Function` и описывающий методы получения границ области определения.

Создайте наследующие от него публичные классы `Sin`, `Cos` и `Tan`, объекты которых вычисляют, соответственно, значения синуса, косинуса и тангенса. Для получения значений следует воспользоваться методами `Math.sin()`, `Math.cos()` и `Math.tan()`.

Задание 4

Создайте пакет `functions.meta`, в нём будут описаны классы функций, позволяющие комбинировать функции.

Создайте класс `Sum`, объекты которого представляют собой функции, являющиеся суммой двух других функций. Класс должен реализовывать интерфейс `Function`. Конструктор класса должен получать ссылки типа `Function` на объекты суммируемых функций, а область определения функции должна получаться как пересечение областей определения исходных функций.

Аналогично, создайте класс `Mult`, объекты которого представляют собой функции, являющиеся произведением двух других функций.

Создайте класс `Power`, объекты которого представляют собой функции, являющиеся степенью другой функции. Конструктор класса должен получать ссылку на объекты базовой функции и степень, в которую должны возводиться её значения. Область определения функции можно считать совпадающей с областью определения исходной функции (хотя математически это не всегда так).

Создайте класс `Scale`, объекты которого описывают функции, полученные из исходных функций путём масштабирования вдоль осей координат. Конструктор класса должен получать ссылку на объект исходной функции, а также коэффициенты масштабирования вдоль оси абсцисс и оси ординат. Область определения функции должна получаться из области определения исходной функции масштабированием вдоль оси абсцисс, а значение функции – масштабированием значения исходной функции вдоль оси ординат.

Коэффициенты масштабирования могут быть отрицательными.

Аналогично, создайте класс `Shift`, объекты которого описывают функции, полученные из исходных функций путём сдвига вдоль осей координат.

Также создайте класс `Composition`, объекты которого описывают композицию двух исходных функций. Конструктор класса должен получать ссылки на объекты первой и второй функции. Область определения функции можно считать совпадающей с областью определения исходной функции (хотя математически это не всегда так).

Задание 5

В пакете `functions` создайте класс `Functions`, содержащий вспомогательные статические методы для работы с функциями. Сделайте так, чтобы в программе вне этого класса нельзя было создать его объект. Класс должен содержать следующие методы:

- `public static Function shift(Function f, double shiftX, double shiftY)` – возвращает объект функции, полученной из исходной сдвигом вдоль осей;
- `public static Function scale(Function f, double scaleX, double scaleY)` – возвращает объект функции, полученной из исходной масштабированием вдоль осей;
- `public static Function power(Function f, double power)` – возвращает объект функции, являющейся заданной степенью исходной;
- `public static Function sum(Function f1, Function f2)` – возвращает объект функции, являющейся суммой двух исходных;
- `public static Function mult(Function f1, Function f2)` – возвращает объект функции, являющейся произведением двух исходных;
- `public static Function composition(Function f1, Function f2)` – возвращает объект функции, являющейся композицией двух исходных.

При написании методов следует воспользоваться созданными ранее классами из пакета `functions.meta`.

Задание 6

В пакете `functions` создайте класс `TabulatedFunctions`, содержащий вспомогательные статические методы для работы с табулированными функциями. Сделайте так, чтобы в программе вне этого класса нельзя было создать его объект.

Опишите в классе метод `public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount)`, получающий функцию и возвращающий её табулированный аналог на заданном отрезке с заданным количеством точек.

Если указанные границы для табулирования выходят за область определения функции, метод должен выбрасывать исключение `IllegalArgumentException`.

Поскольку метод возвращает ссылку интерфейсного типа, можно возвращать

объект любого из классов, реализующих этот интерфейс. В последующих работах в код будет добавлена возможность выбора класса для создания экземпляра.

Задание 7

В класс TabulatedFunctions добавьте следующие методы.

Метод вывода табулированной функции в байтовый поток `public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out)` должен в указанный поток вывести значения, по которым потом можно будет восстановить табулированную функцию, а именно количество точек в ней и значения координат точек.

Метод ввода табулированной функции из байтового потока `public static TabulatedFunction inputTabulatedFunction(InputStream in)` должен считывать из указанного потока данные о табулированной функции, создавать и настраивать её объект и возвращать его из метода.

Метод записи табулированной функции в символьный поток `public static void writeTabulatedFunction(TabulatedFunction function, Writer out)` должен в указанный поток вывести значения, по которым потом можно будет восстановить табулированную функцию, а именно количество точек в ней и значения координат точек. Проще всего считать, что значения записываются в строку и разделяются пробелами.

Метод чтения табулированной функции из символьного потока `public static TabulatedFunction readTabulatedFunction(Reader in)` должен считывать из указанного потока данные о табулированной функции, создавать и настраивать её объект и возвращать его из метода.

При написании методов в первых трёх случаях необходимо воспользоваться потоками-обёртками, облегчающими ввод и вывод данных в требующейся форме, а в четвёртом случае – классом StreamTokenizer.

При написании методов, считающих табулированную функцию, следует считать, что данные в потоке записаны правильные данные (проверку корректности вводимых данных делать не следует).

Поскольку методы ввода и чтения возвращают ссылку интерфейсного типа, можно возвращать объект любого из классов, реализующих этот интерфейс. В последующих работах в код будет добавлена возможность выбора класса для создания экземпляра.

Подумайте и обоснуйте, как следует в этих методах поступить с возникающим исключением IOException.

Подумайте и обоснуйте, следует ли закрывать потоки внутри этих методов.

Задание 8

Проверьте работу написанных классов.

Создайте по одному объекту классов `Sin` и `Cos`, выведите в консоль значения этих функций на отрезке от 0 до π с шагом 0,1.

С помощью метода `TabulatedFunctions.tabulate()` создайте табулированные аналоги этих функций на отрезке от 0 до π с 10 точками. Выведите в консоль значения этих функций на отрезке от 0 до π с шагом 0,1 и сравните со значениями исходных функций.

С помощью методов класса `Functions` создайте объект функции, являющейся суммой квадратов табулированных аналогов синуса и косинуса. Выведите в консоль значения этой функции на отрезке от 0 до π с шагом 0,1. Попробуйте изменять количество точек в табулированных аналогах и исследуйте, как при этом изменяется результирующая функция.

С помощью метода `TabulatedFunctions.tabulate()` создайте табулированный аналог экспоненты на отрезке от 0 до 10 с 11 точками. С помощью метода

`TabulatedFunctions.writeTabulatedFunction()` выведите его в файл. Далее с помощью метода `TabulatedFunctions.readTabulatedFunction()` считайте табулированную функцию из этого файла. Выведите и сравните значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

С помощью метода `TabulatedFunctions.tabulate()` создайте табулированный аналог логарифма по натуральному основанию на отрезке от 0 до 10 с 11 точками. С помощью метода `TabulatedFunctions.outputTabulatedFunction()` выведите его в файл (имя файла должно отличаться от предыдущего случая). Далее с помощью метода `TabulatedFunctions.inputTabulatedFunction()` считайте табулированную функцию из этого файла. Выведите и сравните значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

Изучите содержимое всех получаемых файлов и сделайте выводы о преимуществах и недостатках каждого из форматов хранения.

Задание 9

Сделайте так, чтобы объекты всех классов, реализующих интерфейс `TabulatedFunction`, были сериализуемыми. Для этого рассмотрите два случая:

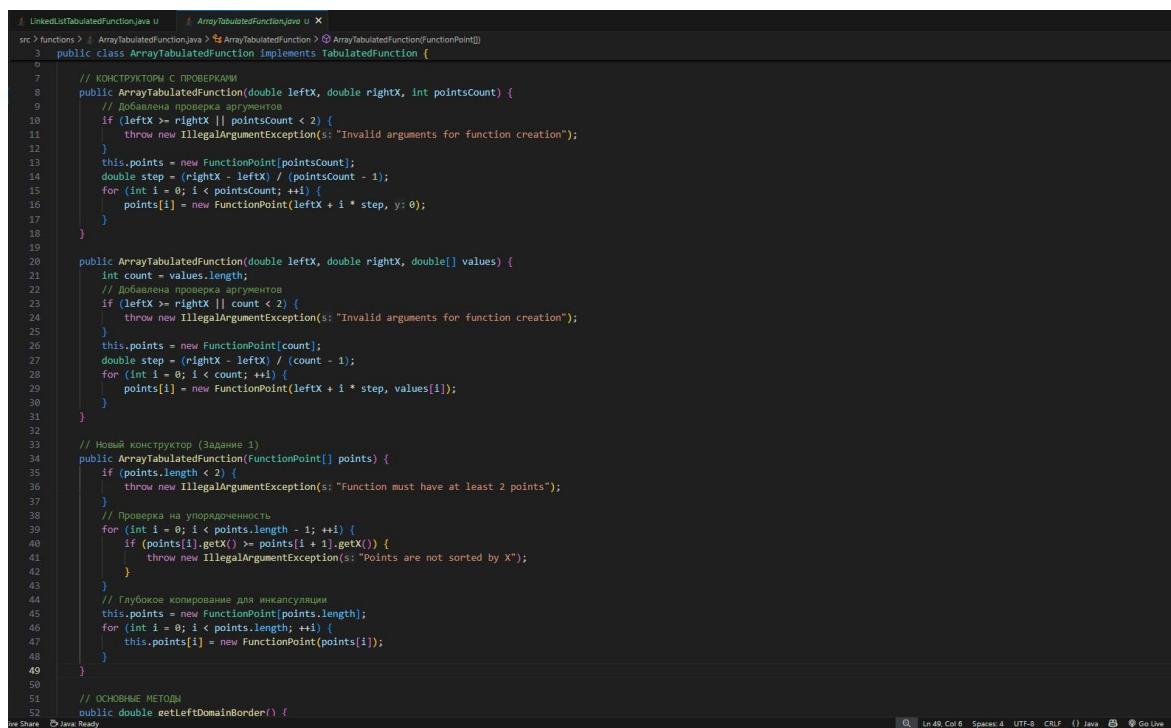
1. с использованием интерфейса `java.io.Serializable`
2. с использованием интерфейса `java.io.Externalizable`

Проверьте работу написанных классов. С помощью метода `TabulatedFunctions.tabulate()` и метода класса `Functions` создайте табулированный аналог логарифма по натуральному основанию, взятого от экспоненты на отрезке от 0 до 10 с 11 точками. Сериализуйте полученный объект в файл (имя файла должно отличаться от предыдущих случаев). Далее десериализуйте табулированную функцию из этого файла. Выведите значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1. Изучите содержимое файлов,

получаемых при реализации механизма сериализации с использованием интерфейса `java.io.Serializable` и при реализации механизма сериализации с использованием интерфейса `java.io.Externalizable`. Сделайте выводы о преимуществах и недостатках каждого из способов.

Задание 1

В классах `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` я добавил конструкторы, получающие сразу все точки функции в виде массива объектов типа `FunctionPoint`. Если точек задано меньше двух, или если точки в массиве не упорядочены по значению абсциссы, конструкторы выбрасывают исключение `IllegalArgumentException`. При написании конструкторов я обеспечил корректную инкапсуляцию.



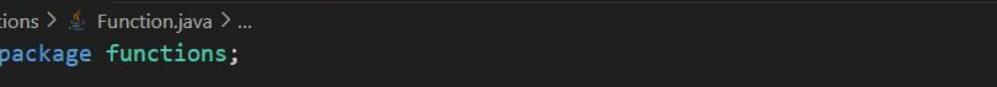
```
# LinkedListTabulatedFunction.java  # ArrayTabulatedFunction.java  #
src > functions >  ArrayTabulatedFunction.java > $ ArrayTabulatedFunction > ArrayTabulatedFunction[FunctionPoint[]]
3  public class ArrayTabulatedFunction implements TabulatedFunction {
0
7   // КОНСТРУКТОРЫ С ПРОВЕРКАМИ
8   public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
9     // Добавлена проверка аргументов
10    if (leftX >= rightX || pointsCount < 2) {
11      throw new IllegalArgumentException("Invalid arguments for function creation");
12    }
13    this.points = new FunctionPoint[pointsCount];
14    double step = (rightX - leftX) / (pointsCount - 1);
15    for (int i = 0; i < pointsCount; ++i) {
16      points[i] = new FunctionPoint(leftX + i * step, y: 0);
17    }
18  }
19
20  public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {
21    int count = values.length;
22    // Добавлена проверка аргументов
23    if (leftX >= rightX || count < 2) {
24      throw new IllegalArgumentException("Invalid arguments for function creation");
25    }
26    this.points = new FunctionPoint[count];
27    double step = (rightX - leftX) / (count - 1);
28    for (int i = 0; i < count; ++i) {
29      points[i] = new FunctionPoint(leftX + i * step, values[i]);
30    }
31  }
32
33  // Новый конструктор (Задание 1)
34  public ArrayTabulatedFunction(FunctionPoint[] points) {
35    if (points.length < 2) {
36      throw new IllegalArgumentException("Function must have at least 2 points");
37    }
38    // Проверка на упорядоченность
39    for (int i = 0; i < points.length - 1; ++i) {
40      if (points[i].getX() >= points[i + 1].getX()) {
41        throw new IllegalArgumentException("Points are not sorted by X");
42      }
43    }
44    // Глубокое копирование для инкапсуляции
45    this.points = new FunctionPoint[points.length];
46    for (int i = 0; i < points.length; ++i) {
47      this.points[i] = new FunctionPoint(points[i]);
48    }
49  }
50
51  // ОСНОВНЫЕ МЕТОДЫ
52  public double getLeftDomainBorder() {
```

Задание 2

В пакете `functions` я создал интерфейс `Function`, описывающий функции одной переменной и содержащий следующие методы:

- `public double getLeftDomainBorder()` – возвращает значение левой границы области определения функции;
 - `public double getRightDomainBorder()` – возвращает значение правой границы области определения функции;
 - `public double getFunctionValue(double x)` – возвращает значение функции в заданной точке.

Я исключил соответствующие методы из интерфейса TabulatedFunction и сделал так, чтобы он расширял интерфейс Function. Теперь табулированные функции будут частным случаем функций одной переменной.



```
src > functions > Function.java > ...
1 package functions;
2
3 public interface Function {
4     double getLeftDomainBorder();
5     double getRightDomainBorder();
6     double getFunctionValue(double x);
7 }
8
```

```
src > functions > TabulatedFunction.java U ...
1 package functions;
2
3 /* 1. Расширяем Function (extends Function)
4 | 2. Добавляем Serializable для Задания 9 */
5 public interface TabulatedFunction extends Function, java.io.Serializable {
6     /* Методы getLeftDomainBorder, getRightDomainBorder, getFunctionValue
7     теперь наследуются из Function. Удаляем их отсюда. */
8
9     int getPointsCount();
10    FunctionPoint getPoint(int index);
11    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;
12    double getPointX(int index);
13    void setPointX(int index, double x) throws InappropriateFunctionPointException;
14    double getPointY(int index);
15    void setPointY(int index, double y);
16    void deletePoint(int index);
17    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
18 }
19
```

Задание 3

Я создал пакет functions.basic, в нём описаны классы ряда функций, заданных аналитически.

Я также создал в пакете публичный класс Exp, объекты которого вычисляют значение экспоненты. Класс реализовывает интерфейс Function. Для вычисления экспоненты я воспользовался методом Math.exp(), а для возвращения значений границ области определения – константами из класса Double.

Аналогично, я создал класс Log, объекты которого вычисляют значение логарифма по заданному основанию. Основание передаётся как параметр конструктора. Для вычисления логарифма я воспользовался методом Math.log().

Я создал класс TrigonometricFunction, реализующий интерфейс Function и описывающий методы получения границ области определения.

Я создайте наследующие от него публичные классы Sin, Cos и Tan, объекты которых вычисляют, соответственно, значения синуса, косинуса и тангенса. Для получения значений я воспользовался методами Math.sin(), Math.cos() и Math.tan().

```
src > functions > basic > Exp.java U ...
1 package functions.basic;
2 import functions.Function;
3
4 public class Exp implements Function {
5     public double getLeftDomainBorder() { return Double.NEGATIVE_INFINITY; }
6     public double getRightDomainBorder() { return Double.POSITIVE_INFINITY; }
7     public double getFunctionValue(double x) { return Math.exp(x); }
8 }
9
```

```
src > functions > basic > Log.java > ...
1 package functions.basic;
2 import functions.Function;
3
4 public class Log implements Function {
5     private double base;
6     public Log(double base) {
7         if (base <= 0 || base == 1) throw new IllegalArgumentException(s: "Invalid base");
8         this.base = base;
9     }
10    public double getLeftDomainBorder() { return 0; } // log(x) определён для x > 0
11    public double getRightDomainBorder() { return Double.POSITIVE_INFINITY; }
12    public double getFunctionValue(double x) { return Math.log(x) / Math.log(this.base); }
13 }
14
```

```
src > functions > basic > TrigonometricFunction.java > ...
1 package functions.basic;
2 import functions.Function;
3
4 public abstract class TrigonometricFunction implements Function {
5     public double getLeftDomainBorder() { return Double.NEGATIVE_INFINITY; }
6     public double getRightDomainBorder() { return Double.POSITIVE_INFINITY; }
7 }
8
```

```
src > functions > basic > Sin.java > ...
1 package functions.basic;
2
3 public class Sin extends TrigonometricFunction {
4     public double getFunctionValue(double x) { return Math.sin(x); }
5 }
6
```

```
src > functions > basic > Cos.java > ...
1 package functions.basic;
2
3 public class Cos extends TrigonometricFunction {
4     public double getFunctionValue(double x) { return Math.cos(x); }
5 }
6
```

```
src > functions > basic > Tan.java > ...
1 package functions.basic;
2
3 public class Tan extends TrigonometricFunction {
4     public double getFunctionValue(double x) { return Math.tan(x); }
5 }
6
```

Задание 4

Я создал пакет functions.meta, в нём описаны классы функций, позволяющие комбинировать функции.

Я также создал класс Sum, объекты которого представляют собой функции, являющиеся суммой двух других функций. Класс реализовывает интерфейс Function. Конструктор класса получает ссылки типа Function на объекты суммируемых функций, а область определения функции получается как пересечение областей определения исходных функций.

Аналогично, я создал класс Mult, объекты которого представляют собой функции, являющиеся произведением двух других функций.

Я создайте класс Power, объекты которого представляют собой функции, являющиеся степенью другой функции. Конструктор класса получает ссылку на объекты базовой функции и степень, в которую возводится её значение. Область определения функции можно считать совпадающей с областью определения исходной функции (хотя математически это не всегда так).

Я создал класс Scale, объекты которого описывают функции, полученные из исходных функций путём масштабирования вдоль осей координат. Конструктор класса получают ссылку на объект исходной функции, а также коэффициенты масштабирования вдоль оси абсцисс и оси ординат. Область определения функции получается из области определения исходной функции масштабированием вдоль оси абсцисс, а значение функции – масштабированием значения исходной функции вдоль оси ординат. Коэффициенты масштабирования могут быть отрицательными.

Аналогично, я создал класс Shift, объекты которого описывают функции, полученные из исходных функций путём сдвига вдоль осей координат.

Также создайте класс Composition, объекты которого описывают композицию двух исходных функций. Конструктор класса получает ссылки на объекты первой и второй функции. Область определения функции можно считать совпадающей с областью определения исходной функции (хотя математически это не всегда так).



```
Sum.java U X Mult.java U Power.java U Scale.java U Shift.java U Composition.java U
src > functions > meta > Sum.java > ...
1 package functions.meta;
2 import functions.Function;
3
4 public class Sum implements Function {
5     private Function f1, f2;
6     private double left, right;
7     public Sum(Function f1, Function f2) {
8         this.f1 = f1; this.f2 = f2;
9         this.left = Math.max(f1.getLeftDomainBorder(), f2.getLeftDomainBorder());
10        this.right = Math.min(f1.getRightDomainBorder(), f2.getRightDomainBorder());
11    }
12    public double getLeftDomainBorder() { return left; }
13    public double getRightDomainBorder() { return right; }
14    public double getFunctionValue(double x) { return f1.getFunctionValue(x) + f2.getFunctionValue(x); }
15 }
16
```



```
Sum.java U Mult.java U X Power.java U Scale.java U Shift.java U Composition.java U
src > functions > meta > Mult.java > ...
1 package functions.meta;
2 import functions.Function;
3
4 public class Mult implements Function {
5     private Function f1, f2;
6     private double left, right;
7     public Mult(Function f1, Function f2) {
8         this.f1 = f1; this.f2 = f2;
9         this.left = Math.max(f1.getLeftDomainBorder(), f2.getLeftDomainBorder());
10        this.right = Math.min(f1.getRightDomainBorder(), f2.getRightDomainBorder());
11    }
12    public double getLeftDomainBorder() { return left; }
13    public double getRightDomainBorder() { return right; }
14    public double getFunctionValue(double x) { return f1.getFunctionValue(x) * f2.getFunctionValue(x); }
15 }
16
```

src > functions > meta > Power.java > ...

```
1 package functions.meta;
2 import functions.Function;
3
4 public class Power implements Function {
5     private Function f;
6     private double power;
7
8     public Power(Function f, double power) {
9         this.f = f;
10        this.power = power;
11    }
12
13    public double getLeftDomainBorder() { return f.getLeftDomainBorder(); }
14    public double getRightDomainBorder() { return f.getRightDomainBorder(); }
15    public double getFunctionValue(double x) {
16        return Math.pow(f.getFunctionValue(x), power);
17    }
18 }
19
```

src > functions > meta > Scale.java > ...

```
1 package functions.meta;
2 import functions.Function;
3
4 public class Scale implements Function {
5     private Function f;
6     private double scaleX, scaleY;
7
8     public Scale(Function f, double scaleX, double scaleY) {
9         this.f = f;
10        this.scaleX = scaleX;
11        this.scaleY = scaleY;
12    }
13
14    public double getLeftDomainBorder() { return f.getLeftDomainBorder() * scaleX; }
15    public double getRightDomainBorder() { return f.getRightDomainBorder() * scaleX; }
16    public double getFunctionValue(double x) {
17        return f.getFunctionValue(x / scaleX) * scaleY;
18    }
19 }
```

```

src > functions > meta > Shift.java > ...
1 package functions.meta;
2 import functions.Function;
3
4 public class Shift implements Function {
5     private Function f;
6     private double shiftX, shiftY;
7
8     public Shift(Function f, double shiftX, double shiftY) {
9         this.f = f;
10        this.shiftX = shiftX;
11        this.shiftY = shiftY;
12    }
13
14    public double getLeftDomainBorder() { return f.getLeftDomainBorder() + shiftX; }
15    public double getRightDomainBorder() { return f.getRightDomainBorder() + shiftX; }
16    public double getFunctionValue(double x) {
17        return f.getFunctionValue(x - shiftX) + shiftY;
18    }
19 }

```



```

src > functions > meta > Composition.java > ...
1 package functions.meta;
2 import functions.Function;
3
4 public class Composition implements Function {
5     private Function f1, f2;
6
7     public Composition(Function f1, Function f2) {
8         this.f1 = f1;
9         this.f2 = f2;
10    }
11
12    public double getLeftDomainBorder() { return f1.getLeftDomainBorder(); }
13    public double getRightDomainBorder() { return f1.getRightDomainBorder(); }
14    public double getFunctionValue(double x) {
15        // f1(f2(x))
16        return f1.getFunctionValue(f2.getFunctionValue(x));
17    }
18 }
19

```

Задание 5 В пакете functions я создал класс Functions, содержащий вспомогательные статические методы для работы с функциями. Я сделал так, что в программе вне этого класса нельзя создать его объект. Класс содержит следующие методы:

- public static Function shift(Function f, double shiftX, double shiftY) – возвращает объект функции, полученной из исходной сдвигом вдоль осей;
- public static Function scale(Function f, double scaleX, double scaleY) – возвращает объект функции, полученной из исходной масштабированием вдоль осей;
- public static Function power(Function f, double power) – возвращает объект функции, являющейся заданной степенью исходной;
- public static Function sum(Function f1, Function f2) – возвращает объект функции, являющейся суммой двух исходных;
- public static Function mult(Function f1, Function f2) – возвращает объект функции, являющейся произведением двух исходных;
- public static Function composition(Function f1, Function f2) – возвращает объект функции, являющейся композицией двух исходных.

При написании методов я воспользовался созданными ранее классами из пакета functions.meta.

```
src > functions > Functions.java > ...
1 package functions;
2 import functions.meta.*;
3
4 public class Functions {
5     // Приватный конструктор, чтобы нельзя было создать объект
6     private Functions() {}
7
8     public static Function shift(Function f, double shiftX, double shiftY) {
9         return new Shift(f, shiftX, shiftY);
10    }
11    public static Function scale(Function f, double scaleX, double scaleY) {
12        return new Scale(f, scaleX, scaleY);
13    }
14    public static Function power(Function f, double power) {
15        return new Power(f, power);
16    }
17    public static Function sum(Function f1, Function f2) {
18        return new Sum(f1, f2);
19    }
20    public static Function mult(Function f1, Function f2) {
21        return new Mult(f1, f2);
22    }
23    public static Function composition(Function f1, Function f2) {
24        return new Composition(f1, f2);
25    }
26 }
27 }
```

Задание 6

В пакете functions я создал класс TabulatedFunctions, содержащий вспомогательные статические методы для работы с табулированными функциями. Я сделал так, что в программе вне этого класса нельзя создать его объект.

Я описал в классе метод public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount), получающий функцию и возвращающий её табулированный аналог на заданном отрезке с заданным количеством точек.

Если указанные границы для табулирования выходят за область определения функции, метод выбрасывает исключение IllegalArgumentException.

Поскольку метод возвращает ссылку интерфейсного типа, можно возвращать объект любого из классов, реализующих этот интерфейс. В последующих работах в код будет добавлена возможность выбора класса для создания экземпляра.

```
src > functions > TabulatedFunctions.java > ...
1 package functions;
2 import java.io.*;
3
4 public class TabulatedFunctions {
5     // Приватный конструктор
6     private TabulatedFunctions() {}
7
8     // --- Метод для Задания 6 ---
9     public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
10        if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
11            throw new IllegalArgumentException(s: "Tabulation bounds are out of function's domain");
12        }
13        if (pointsCount < 2) {
14            throw new IllegalArgumentException(s: "Points count must be 2 or more");
15        }
16
17        FunctionPoint[] points = new FunctionPoint[pointsCount];
18        double step = (rightX - leftX) / (pointsCount - 1);
19        for (int i = 0; i < pointsCount; i++) {
20            double x = leftX + i * step;
21            points[i] = new FunctionPoint(x, function.getFunctionValue(x));
22        }
23        // Возвращаем объект одного из классов, реализующих интерфейс
24        return new ArrayTabulatedFunction(points);
25        // Можно вернуть и LinkedListTabulatedFunction, по заданию это не важно
26    }
27 }
28 }
```

Задание 7

В класс TabulatedFunctions я добавил следующие методы.

Метод вывода табулированной функции в байтовый поток public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) в указанный поток выводит значения, по которым потом можно восстановить табулированную функцию, а именно количество точек в ней и значения координат точек.

Метод ввода табулированной функции из байтового потока public static TabulatedFunction inputTabulatedFunction(InputStream in) считывает из указанного потока данные о табулированной функции, создаёт и настраивает её объект и возвращает его из метода.

Метод записи табулированной функции в символьный поток public static void writeTabulatedFunction(TabulatedFunction function, Writer out) в указанный поток выводит значения, по которым потом можно восстановить табулированную функцию, а именно количество точек в ней и значения координат точек. Проще всего считать, что значения записываются в строку и разделяются пробелами.

Метод чтения табулированной функции из символьного потока public static TabulatedFunction readTabulatedFunction(Reader in) считывает из указанного потока данные о табулированной функции, создаёт и настраивает её объект и возвращает его из метода. При написании методов в первых трёх случаях я воспользовался потоками-обёртками, облегчающими ввод и вывод данных в требующейся форме, а в четвёртом случае – классом StreamTokenizer.

При написании методов, считающих табулированную функцию, я считал, что в потоке записаны правильные данные (проверку корректности вводимых данных делать не делал).

Поскольку методы ввода и чтения возвращают ссылку интерфейсного типа, можно возвращать объект любого из классов, реализующих этот интерфейс. В последующих работах в код будет добавлена возможность выбора класса для создания экземпляра. Я подумал и обосновал, как следует в этих методах поступить с возникающим исключением IOException.

Я подумал и обосновал, следует ли закрывать потоки внутри этих методов.

Что я сделал и как обосновал:

В методах для ввода/вывода (таких как outputTabulatedFunction, inputTabulatedFunction и т.д.) исключение IOException не обрабатывается (через try-catch), а прорывается наверх (с помощью throws IOException).

Обоснование: Эти методы являются утилитарными, то есть "инструментами". Они не обладают информацией о том, насколько критична ошибка ввода-вывода (например, "файл не найден" или "диск переполнен").

- Если бы метод "проглотил" исключение, вызвавший его код (например, main) не узнал бы, что операция провалилась.
- Если бы метод сам обработал исключение, он не знал бы, как правильно это сделать (попробовать еще раз? сообщить пользователю? завершить программу?).

Пробрасывая исключение, метод передает ответственность за принятие решения тому, кто его вызвал. Вызывающий код (`main`) имеет полный контекст и может корректно отреагировать на ошибку.

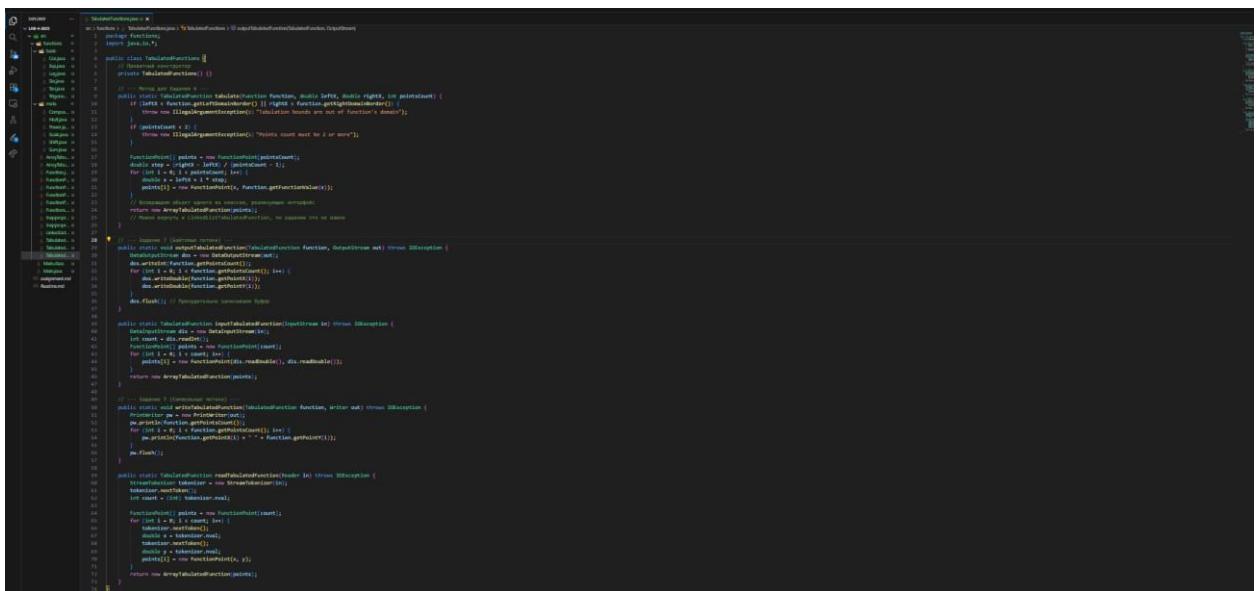
Обоснование по закрытию потоков

Методы ввода/вывода не закрывают потоки (`InputStream`, `OutputStream`, `Writer`, `Reader`), которые они получают в качестве параметров.

Обоснование: В Java принято правило: "Кто создал (открыл) поток, тот его и закрывает".

Эти методы не создают потоки, а получают их “в пользование” от другого кода (в нашем случае — от `main`, который создает `new FileWriter(...)`).

- Если бы наш метод-инструмент закрыл поток, то вызывающий код (`main`) не смог бы, например, записать в этот же файл что-то еще после завершения нашего метода.
 - Ответственность за закрытие потока (`.close()`) лежит на том коде, который его открыл (`main`), и который точно знает, когда этот поток больше не понадобится.



Задание 8

Я проверил работу написанных классов.

Я создал по одному объекту классов Sin и Cos, вывел в консоль значения этих функций на отрезке от 0 до π с шагом 0,1.

С помощью метода TabulatedFunctions.tabulate() я создал табулированные аналоги этих функций на отрезке от 0 до π с 10 точками. Вывел в консоль значения этих функций на отрезке от 0 до π с шагом 0,1 и сравнил со значениями исходных функций. С помощью методов класса Functions я создал объект функции, являющейся суммой квадратов табулированных аналогов синуса и косинуса. Вывел в консоль значения этой функций на отрезке от 0 до π с шагом 0,1. Я попробовал изменять количество точек в табулированных аналогах и исследовал, как при этом изменяется результирующая функция.

С помощью метода TabulatedFunctions.tabulate() я создал табулированный аналог экспоненты на отрезке от 0 до 10 с 11 точками. С помощью метода TabulatedFunctions.writeTabulatedFunction() вывел его в файл. Далее с помощью метода TabulatedFunctions.readTabulatedFunction() считал табулированную функцию из этого файла. Вывел и сравнил значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

С помощью метода TabulatedFunctions.tabulate() я создал табулированный аналог логарифма по натуральному основанию на отрезке от 0 до 10 с 11 точками. С помощью метода TabulatedFunctions.outputTabulatedFunction() вывел его в файл (имя файла отличается от предыдущего случая). Далее с помощью метода TabulatedFunctions.inputTabulatedFunction() я считал табулированную функцию из этого файла. Вывел и сравнил значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

Я изучите содержимое всех получаемых файлов и сделал выводы о преимуществах и недостатках каждого из форматов хранения.

Выводы:

При выполнении работы были созданы два файла для одной и той же функции: tabulatedexp.txt (символьный поток) и tabulated-log.dat (байтовый поток).

1. Символьный поток (write.../read..., файл .txt)
 - Преимущества: Файл является текстовым (tabulated-exp.txt), его можно открыть в любом редакторе (Блокнот, VS Code) и прочитать. Он понятен человеку и его легко можно исправить вручную.
 - Недостатки: Очень неэффективный. Файл занимает много места на диске, так как числа (например, 3.14159) хранятся как последовательность символов, а не как 8 байт. Процесс записи и чтения медленный, так как требует постоянной конвертации чисел в строки и обратно.
2. Байтовый поток (output.../input..., файл .dat)
 - Преимущества: Очень эффективный. Файл (tabulated-log.dat) занимает значительно меньше места, так как числа double записываются напрямую в их 8-байтовом бинарном представлении. Чтение и запись происходят очень быстро, так как отсутствует этап конвертации.
 - Недостатки: Файл абсолютно нечитаем для человека. Его нельзя открыть в Блокноте и понять, что в нём содержится.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Program Files\Java\jdk-20\bin\java.exe -agentlib:jdwp=transport=dt_socket,server=y,address=localhost:5099 -Xt-show-command-lineexception-messages -cp "C:\Users\louis\Videos\Monitoring\Code\user\workspaceStorage\327652\327651\src\main\java\com\louis\test\ad\2025\2025_2025\Talbunarycosine.java"

x=0, y=0, sin(x)=0.0000, cos(x)=1.0000
x=0, y=1, sin(x)=0.8415, cos(x)=0.5403
x=0, y=2, sin(x)=0.3827, cos(x)=0.9239
x=0, y=3, sin(x)=0.2955, cos(x)=0.9553
x=0, y=4, sin(x)=0.1988, cos(x)=0.9811
x=0, y=5, sin(x)=0.0985, cos(x)=0.9955
x=0, y=6, sin(x)=0.0566, cos(x)=0.9925
x=0, y=7, sin(x)=0.0162, cos(x)=0.9964
x=0, y=8, sin(x)=0.0000, cos(x)=0.9999
x=0, y=9, sin(x)=0.0000, cos(x)=0.9999
x=1, y=0, sin(x)=0.8415, cos(x)=0.5403
x=1, y=1, sin(x)=0.3827, cos(x)=0.9239
x=1, y=2, sin(x)=0.2955, cos(x)=0.9553
x=1, y=3, sin(x)=0.1988, cos(x)=0.9811
x=1, y=4, sin(x)=0.0985, cos(x)=0.9955
x=1, y=5, sin(x)=0.0566, cos(x)=0.9925
x=1, y=6, sin(x)=0.0162, cos(x)=0.9964
x=1, y=7, sin(x)=0.0000, cos(x)=0.9999
x=1, y=8, sin(x)=0.0000, cos(x)=0.9999
x=1, y=9, sin(x)=0.0000, cos(x)=0.9999
... Summary: 18 errors, 1 warning

x=0, y=0, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=1, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=2, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=3, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=4, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=5, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=6, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=7, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=8, Talbunarycosine Sin & Cos (18 errors) ...
x=0, y=9, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=0, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=1, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=2, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=3, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=4, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=5, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=6, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=7, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=8, Talbunarycosine Sin & Cos (18 errors) ...
x=1, y=9, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=0, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=1, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=2, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=3, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=4, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=5, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=6, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=7, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=8, Talbunarycosine Sin & Cos (18 errors) ...
x=2, y=9, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=0, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=1, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=2, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=3, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=4, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=5, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=6, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=7, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=8, Talbunarycosine Sin & Cos (18 errors) ...
x=3, y=9, Talbunarycosine Sin & Cos (18 errors) ...
... Summary: 18 errors, 1 warning

x=0, y=0, Sin(x)*cos(x) = 0.0000
x=0, y=1, Sin(x)*cos(x) = 0.0000
x=0, y=2, Sin(x)*cos(x) = 0.0000
x=0, y=3, Sin(x)*cos(x) = 0.0000
x=0, y=4, Sin(x)*cos(x) = 0.0000
x=0, y=5, Sin(x)*cos(x) = 0.0000
x=0, y=6, Sin(x)*cos(x) = 0.0000
x=0, y=7, Sin(x)*cos(x) = 0.0000
x=0, y=8, Sin(x)*cos(x) = 0.0000
x=0, y=9, Sin(x)*cos(x) = 0.0000
x=1, y=0, Sin(x)*cos(x) = 0.0000
x=1, y=1, Sin(x)*cos(x) = 0.0000
x=1, y=2, Sin(x)*cos(x) = 0.0000
x=1, y=3, Sin(x)*cos(x) = 0.0000
x=1, y=4, Sin(x)*cos(x) = 0.0000
x=1, y=5, Sin(x)*cos(x) = 0.0000
x=1, y=6, Sin(x)*cos(x) = 0.0000
x=1, y=7, Sin(x)*cos(x) = 0.0000
x=1, y=8, Sin(x)*cos(x) = 0.0000
x=1, y=9, Sin(x)*cos(x) = 0.0000
x=2, y=0, Sin(x)*cos(x) = 0.0000
x=2, y=1, Sin(x)*cos(x) = 0.0000
x=2, y=2, Sin(x)*cos(x) = 0.0000
x=2, y=3, Sin(x)*cos(x) = 0.0000
x=2, y=4, Sin(x)*cos(x) = 0.0000
x=2, y=5, Sin(x)*cos(x) = 0.0000
x=2, y=6, Sin(x)*cos(x) = 0.0000
x=2, y=7, Sin(x)*cos(x) = 0.0000
x=2, y=8, Sin(x)*cos(x) = 0.0000
x=2, y=9, Sin(x)*cos(x) = 0.0000
x=3, y=0, Sin(x)*cos(x) = 0.0000
x=3, y=1, Sin(x)*cos(x) = 0.0000
x=3, y=2, Sin(x)*cos(x) = 0.0000
x=3, y=3, Sin(x)*cos(x) = 0.0000
x=3, y=4, Sin(x)*cos(x) = 0.0000
x=3, y=5, Sin(x)*cos(x) = 0.0000
x=3, y=6, Sin(x)*cos(x) = 0.0000
x=3, y=7, Sin(x)*cos(x) = 0.0000
x=3, y=8, Sin(x)*cos(x) = 0.0000
x=3, y=9, Sin(x)*cos(x) = 0.0000
... Summary: 18 errors, 1 warning

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Projects\Lab-4-2025 & C:\Program Files\Java\jdk-24\bin\java.exe -agentlib:jmap=transport=tcp,socket,server,suspend,y,address=localhost:62897 -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\Baron\ApkData\Learning\Code\Java\src\spaceStorage\src\main\java\Lab-4-2025\exp\*"
... Задание 8.1. Сумма квадратов (sin(x2 + cos2) ...  

x<0, sin2(x<0)+cos2(x<0)  

x<1, sin2(x<1)+cos2(x<1)  

x<2, sin2(x<2)+cos2(x<2)  

x<3, sin2(x<3)+cos2(x<3)  

x<4, sin2(x<4)+cos2(x<4)  

x<5, sin2(x<5)+cos2(x<5)  

x<6, sin2(x<6)+cos2(x<6)  

x<7, sin2(x<7)+cos2(x<7)  

x<8, sin2(x<8)+cos2(x<8)  

x<9, sin2(x<9)+cos2(x<9)  

x<10, sin2(x<10)+cos2(x<10)  

x<11, sin2(x<11)+cos2(x<11)  

x<12, sin2(x<12)+cos2(x<12)  

x<13, sin2(x<13)+cos2(x<13)  

x<14, sin2(x<14)+cos2(x<14)  

x<15, sin2(x<15)+cos2(x<15)  

x<16, sin2(x<16)+cos2(x<16)  

x<17, sin2(x<17)+cos2(x<17)  

x<18, sin2(x<18)+cos2(x<18)  

x<19, sin2(x<19)+cos2(x<19)  

x<20, sin2(x<20)+cos2(x<20)  

x<21, sin2(x<21)+cos2(x<21)  

x<22, sin2(x<22)+cos2(x<22)  

x<23, sin2(x<23)+cos2(x<23)  

x<24, sin2(x<24)+cos2(x<24)  

x<25, sin2(x<25)+cos2(x<25)  

x<26, sin2(x<26)+cos2(x<26)  

x<27, sin2(x<27)+cos2(x<27)  

x<28, sin2(x<28)+cos2(x<28)  

x<29, sin2(x<29)+cos2(x<29)  

x<30, sin2(x<30)+cos2(x<30)  

x<31, sin2(x<31)+cos2(x<31)  

x<32, sin2(x<32)+cos2(x<32)  

x<33, sin2(x<33)+cos2(x<33)  

x<34, sin2(x<34)+cos2(x<34)  

x<35, sin2(x<35)+cos2(x<35)  

x<36, sin2(x<36)+cos2(x<36)  

x<37, sin2(x<37)+cos2(x<37)  

x<38, sin2(x<38)+cos2(x<38)  

x<39, sin2(x<39)+cos2(x<39)  

x<40, sin2(x<40)+cos2(x<40)  

x<41, sin2(x<41)+cos2(x<41)  

x<42, sin2(x<42)+cos2(x<42)  

x<43, sin2(x<43)+cos2(x<43)  

x<44, sin2(x<44)+cos2(x<44)  

x<45, sin2(x<45)+cos2(x<45)  

x<46, sin2(x<46)+cos2(x<46)  

x<47, sin2(x<47)+cos2(x<47)  

x<48, sin2(x<48)+cos2(x<48)  

x<49, sin2(x<49)+cos2(x<49)  

x<50, sin2(x<50)+cos2(x<50)  

x<51, sin2(x<51)+cos2(x<51)  

x<52, sin2(x<52)+cos2(x<52)  

x<53, sin2(x<53)+cos2(x<53)  

x<54, sin2(x<54)+cos2(x<54)  

x<55, sin2(x<55)+cos2(x<55)  

x<56, sin2(x<56)+cos2(x<56)  

x<57, sin2(x<57)+cos2(x<57)  

x<58, sin2(x<58)+cos2(x<58)  

x<59, sin2(x<59)+cos2(x<59)  

x<60, sin2(x<60)+cos2(x<60)  

x<61, sin2(x<61)+cos2(x<61)  

x<62, sin2(x<62)+cos2(x<62)  

x<63, sin2(x<63)+cos2(x<63)  

x<64, sin2(x<64)+cos2(x<64)  

x<65, sin2(x<65)+cos2(x<65)  

x<66, sin2(x<66)+cos2(x<66)  

x<67, sin2(x<67)+cos2(x<67)  

x<68, sin2(x<68)+cos2(x<68)  

x<69, sin2(x<69)+cos2(x<69)  

x<70, sin2(x<70)+cos2(x<70)  

x<71, sin2(x<71)+cos2(x<71)  

x<72, sin2(x<72)+cos2(x<72)  

x<73, sin2(x<73)+cos2(x<73)  

x<74, sin2(x<74)+cos2(x<74)  

x<75, sin2(x<75)+cos2(x<75)  

x<76, sin2(x<76)+cos2(x<76)  

x<77, sin2(x<77)+cos2(x<77)  

x<78, sin2(x<78)+cos2(x<78)  

x<79, sin2(x<79)+cos2(x<79)  

x<80, sin2(x<80)+cos2(x<80)  

x<81, sin2(x<81)+cos2(x<81)  

x<82, sin2(x<82)+cos2(x<82)  

x<83, sin2(x<83)+cos2(x<83)  

x<84, sin2(x<84)+cos2(x<84)  

x<85, sin2(x<85)+cos2(x<85)  

x<86, sin2(x<86)+cos2(x<86)  

x<87, sin2(x<87)+cos2(x<87)  

x<88, sin2(x<88)+cos2(x<88)  

x<89, sin2(x<89)+cos2(x<89)  

x<90, sin2(x<90)+cos2(x<90)  

x<91, sin2(x<91)+cos2(x<91)  

x<92, sin2(x<92)+cos2(x<92)  

x<93, sin2(x<93)+cos2(x<93)  

x<94, sin2(x<94)+cos2(x<94)  

x<95, sin2(x<95)+cos2(x<95)  

x<96, sin2(x<96)+cos2(x<96)  

x<97, sin2(x<97)+cos2(x<97)  

x<98, sin2(x<98)+cos2(x<98)  

x<99, sin2(x<99)+cos2(x<99)  

x<100, sin2(x<100)+cos2(x<100)  

... Задание 8.4. Тест write/read (текстовый файл) ...  

Создано 1 текстовый файл с помощью из выше функции:  

x<0, 0.0, Read(x<0)=0.0  

x<1, 0.1, Read(x<1)=0.1  

x<2, 0.2, Read(x<2)=0.2  

x<3, 0.3, Read(x<3)=0.3  

x<4, 0.4, Read(x<4)=0.4  

x<5, 0.5, Read(x<5)=0.5  

x<6, 0.6, Read(x<6)=0.6  

x<7, 0.7, Read(x<7)=0.7  

x<8, 0.8, Read(x<8)=0.8  

x<9, 0.9, Read(x<9)=0.9  

x<10, 0.1, Read(x<10)=0.1  

x<11, 0.2, Read(x<11)=0.2  

x<12, 0.3, Read(x<12)=0.3  

x<13, 0.4, Read(x<13)=0.4  

x<14, 0.5, Read(x<14)=0.5  

x<15, 0.6, Read(x<15)=0.6  

x<16, 0.7, Read(x<16)=0.7  

x<17, 0.8, Read(x<17)=0.8  

x<18, 0.9, Read(x<18)=0.9  

x<19, 0.1, Read(x<19)=0.1  

x<20, 0.2, Read(x<20)=0.2  

x<21, 0.3, Read(x<21)=0.3  

x<22, 0.4, Read(x<22)=0.4  

x<23, 0.5, Read(x<23)=0.5  

x<24, 0.6, Read(x<24)=0.6  

x<25, 0.7, Read(x<25)=0.7  

x<26, 0.8, Read(x<26)=0.8  

x<27, 0.9, Read(x<27)=0.9  

x<28, 0.1, Read(x<28)=0.1  

x<29, 0.2, Read(x<29)=0.2  

x<30, 0.3, Read(x<30)=0.3  

x<31, 0.4, Read(x<31)=0.4  

x<32, 0.5, Read(x<32)=0.5  

x<33, 0.6, Read(x<33)=0.6  

x<34, 0.7, Read(x<34)=0.7  

x<35, 0.8, Read(x<35)=0.8  

x<36, 0.9, Read(x<36)=0.9  

x<37, 0.1, Read(x<37)=0.1  

x<38, 0.2, Read(x<38)=0.2  

x<39, 0.3, Read(x<39)=0.3  

x<40, 0.4, Read(x<40)=0.4  

x<41, 0.5, Read(x<41)=0.5  

x<42, 0.6, Read(x<42)=0.6  

x<43, 0.7, Read(x<43)=0.7  

x<44, 0.8, Read(x<44)=0.8  

x<45, 0.9, Read(x<45)=0.9  

x<46, 0.1, Read(x<46)=0.1  

x<47, 0.2, Read(x<47)=0.2  

x<48, 0.3, Read(x<48)=0.3  

x<49, 0.4, Read(x<49)=0.4  

x<50, 0.5, Read(x<50)=0.5  

x<51, 0.6, Read(x<51)=0.6  

x<52, 0.7, Read(x<52)=0.7  

x<53, 0.8, Read(x<53)=0.8  

x<54, 0.9, Read(x<54)=0.9  

x<55, 0.1, Read(x<55)=0.1  

x<56, 0.2, Read(x<56)=0.2  

x<57, 0.3, Read(x<57)=0.3  

x<58, 0.4, Read(x<58)=0.4  

x<59, 0.5, Read(x<59)=0.5  

x<60, 0.6, Read(x<60)=0.6  

x<61, 0.7, Read(x<61)=0.7  

x<62, 0.8, Read(x<62)=0.8  

x<63, 0.9, Read(x<63)=0.9  

x<64, 0.1, Read(x<64)=0.1  

x<65, 0.2, Read(x<65)=0.2  

x<66, 0.3, Read(x<66)=0.3  

x<67, 0.4, Read(x<67)=0.4  

x<68, 0.5, Read(x<68)=0.5  

x<69, 0.6, Read(x<69)=0.6  

x<70, 0.7, Read(x<70)=0.7  

x<71, 0.8, Read(x<71)=0.8  

x<72, 0.9, Read(x<72)=0.9  

x<73, 0.1, Read(x<73)=0.1  

x<74, 0.2, Read(x<74)=0.2  

x<75, 0.3, Read(x<75)=0.3  

x<76, 0.4, Read(x<76)=0.4  

x<77, 0.5, Read(x<77)=0.5  

x<78, 0.6, Read(x<78)=0.6  

x<79, 0.7, Read(x<79)=0.7  

x<80, 0.8, Read(x<80)=0.8  

x<81, 0.9, Read(x<81)=0.9  

x<82, 0.1, Read(x<82)=0.1  

x<83, 0.2, Read(x<83)=0.2  

x<84, 0.3, Read(x<84)=0.3  

x<85, 0.4, Read(x<85)=0.4  

x<86, 0.5, Read(x<86)=0.5  

x<87, 0.6, Read(x<87)=0.6  

x<88, 0.7, Read(x<88)=0.7  

x<89, 0.8, Read(x<89)=0.8  

x<90, 0.9, Read(x<90)=0.9  

x<91, 0.1, Read(x<91)=0.1  

x<92, 0.2, Read(x<92)=0.2  

x<93, 0.3, Read(x<93)=0.3  

x<94, 0.4, Read(x<94)=0.4  

x<95, 0.5, Read(x<95)=0.5  

x<96, 0.6, Read(x<96)=0.6  

x<97, 0.7, Read(x<97)=0.7  

x<98, 0.8, Read(x<98)=0.8  

x<99, 0.9, Read(x<99)=0.9  

x<100, 0.1, Read(x<100)=0.1  

... Задание 8.5. Тест output/input (текущий файл) ...  

Создано 1 текстовый файл с помощью из выше функции:  

x<0, 0.0, Read(x<0)=0.0  

x<1, 0.1, Read(x<1)=0.1  

x<2, 0.2, Read(x<2)=0.2  

x<3, 0.3, Read(x<3)=0.3  

x<4, 0.4, Read(x<4)=0.4  

x<5, 0.5, Read(x<5)=0.5  

x<6, 0.6, Read(x<6)=0.6  

x<7, 0.7, Read(x<7)=0.7  

x<8, 0.8, Read(x<8)=0.8  

x<9, 0.9, Read(x<9)=0.9  

x<10, 0.1, Read(x<10)=0.1  

x<11, 0.2, Read(x<11)=0.2  

x<12, 0.3, Read(x<12)=0.3  

x<13, 0.4, Read(x<13)=0.4  

x<14, 0.5, Read(x<14)=0.5  

x<15, 0.6, Read(x<15)=0.6  

x<16, 0.7, Read(x<16)=0.7  

x<17, 0.8, Read(x<17)=0.8  

x<18, 0.9, Read(x<18)=0.9  

x<19, 0.1, Read(x<19)=0.1  

x<20, 0.2, Read(x<20)=0.2  

x<21, 0.3, Read(x<21)=0.3  

x<22, 0.4, Read(x<22)=0.4  

x<23, 0.5, Read(x<23)=0.5  

x<24, 0.6, Read(x<24)=0.6  

x<25, 0.7, Read(x<25)=0.7  

x<26, 0.8, Read(x<26)=0.8  

x<27, 0.9, Read(x<27)=0.9  

x<28, 0.1, Read(x<28)=0.1  

x<29, 0.2, Read(x<29)=0.2  

x<30, 0.3, Read(x<30)=0.3  

x<31, 0.4, Read(x<31)=0.4  

x<32, 0.5, Read(x<32)=0.5  

x<33, 0.6, Read(x<33)=0.6  

x<34, 0.7, Read(x<34)=0.7  

x<35, 0.8, Read(x<35)=0.8  

x<36, 0.9, Read(x<36)=0.9  

x<37, 0.1, Read(x<37)=0.1  

x<38, 0.2, Read(x<38)=0.2  

x<39, 0.3, Read(x<39)=0.3  

x<40, 0.4, Read(x<40)=0.4  

x<41, 0.5, Read(x<41)=0.5  

x<42, 0.6, Read(x<42)=0.6  

x<43, 0.7, Read(x<43)=0.7  

x<44, 0.8, Read(x<44)=0.8  

x<45, 0.9, Read(x<45)=0.9  

x<46, 0.1, Read(x<46)=0.1  

x<47, 0.2, Read(x<47)=0.2  

x<48, 0.3, Read(x<48)=0.3  

x<49, 0.4, Read(x<49)=0.4  

x<50, 0.5, Read(x<50)=0.5  

x<51, 0.6, Read(x<51)=0.6  

x<52, 0.7, Read(x<52)=0.7  

x<53, 0.8, Read(x<53)=0.8  

x<54, 0.9, Read(x<54)=0.9  

x<55, 0.1, Read(x<55)=0.1  

x<56, 0.2, Read(x<56)=0.2  

x<57, 0.3, Read(x<57)=0.3  

x<58, 0.4, Read(x<58)=0.4  

x<59, 0.5, Read(x<59)=0.5  

x<60, 0.6, Read(x<60)=0.6  

x<61, 0.7, Read(x<61)=0.7  

x<62, 0.8, Read(x<62)=0.8  

x<63, 0.9, Read(x<63)=0.9  

x<64, 0.1, Read(x<64)=0.1  

x<65, 0.2, Read(x<65)=0.2  

x<66, 0.3, Read(x<66)=0.3  

x<67, 0.4, Read(x<67)=0.4  

x<68, 0.5, Read(x<68)=0.5  

x<69, 0.6, Read(x<69)=0.6  

x<70, 0.7, Read(x<70)=0.7  

x<71, 0.8, Read(x<71)=0.8  

x<72, 0.9, Read(x<72)=0.9  

x<73, 0.1, Read(x<73)=0.1  

x<74, 0.2, Read(x<74)=0.2  

x<75, 0.3, Read(x<75)=0.3  

x<76, 0.4, Read(x<76)=0.4  

x<77, 0.5, Read(x<77)=0.5  

x<78, 0.6, Read(x<78)=0.6  

x<79, 0.7, Read(x<79)=0.7  

x<80, 0.8, Read(x<80)=0.8  

x<81, 0.9, Read(x<81)=0.9  

x<82, 0.1, Read(x<82)=0.1  

x<83, 0.2, Read(x<83)=0.2  

x<84, 0.3, Read(x<84)=0.3  

x<85, 0.4, Read(x<85)=0.4  

x<86, 0.5, Read(x<86)=0.5  

x<87, 0.6, Read(x<87)=0.6  

x<88, 0.7, Read(x<88)=0.7  

x<89, 0.8, Read(x<89)=0.8  

x<90, 0.9, Read(x<90)=0.9  

x<91, 0.1, Read(x<91)=0.1  

x<92, 0.2, Read(x<92)=0.2  

x<93, 0.3, Read(x<93)=0.3  

x<94, 0.4, Read(x<94)=0.4  

x<95, 0.5, Read(x<95)=0.5  

x<96, 0.6, Read(x<96)=0.6  

x<97, 0.7, Read(x<97)=0.7  

x<98, 0.8, Read(x<98)=0.8  

x<99, 0.9, Read(x<99)=0.9  

x<100, 0.1, Read(x<100)=0.1

```

```

Main.java U tabulated-exp.txt U X
tabulated-exp.txt U X
1 11
2 0.0 1.0
3 1.0 2.718281828459045
4 2.0 7.38905609893065
5 3.0 20.085536923187668
6 4.0 54.598150033144236
7 5.0 148.4131591025766
8 6.0 403.4287934927351
9 7.0 1096.6331584284585
10 8.0 2980.9579870417283
11 9.0 8103.083927575384
12 10.0 22026.465794806718
13

```

Задание 9

Я сделайте так, чтобы объекты всех классов, реализующих интерфейс TabulatedFunction, были сериализуемыми.

Для этого я рассмотрел два случая:

1. с использованием интерфейса java.io.Serializable
2. с использованием интерфейса java.io.Externalizable

Я проверьте работу написанных классов. С помощью

метода TabulatedFunctions.tabulate() и метода класса Functions я создал табулированный аналог логарифма по натуральному основанию, взятого от экспоненты на отрезке от 0 до 10 с 11 точками. Я сериализовал полученный объект в файл (имя файла отличается от предыдущих случаев). Далее я десериализовал табулированную функцию из этого файла. Я вывел значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1. Я изучите содержимое файлов, получаемых при реализации механизма сериализации с использованием интерфейса java.io.Serializable и при реализации механизма сериализации с использованием интерфейса java.io.Externalizable. Я сделайте выводы о преимуществах и недостатках каждого из способов.

Выводы:

В работе было реализовано два подхода к сериализации: Serializable (для ArrayTabulatedFunction) и Externalizable (для LinkedListTabulatedFunction).

1. Serializable (стандартный механизм Java)
 - Преимущества: Невероятно прост в реализации. Достаточно добавить implements Serializable и serialVersionUID, и Java сделает всё остальное сама.
 - Недостатки: Файл (serialized-function.ser) нечитаем для человека и часто занимает *больше* места, чем даже байтовый поток, так как Java сохраняет в

него много служебной информации (полное имя класса, имена полей, иерархию наследования). Механизм "хрупкий" — если изменить класс (например, добавить поле), то прочитать старые сохраненные файлы уже не получится.

2. Externalizable (ручной контроль)

- Преимущества: Обеспечивает полный контроль над процессом сериализации. Файл может быть более компактным, чем при использовании Serializable, поскольку мы сами решаем, какие поля сохранять. Это делает механизм устойчивым к изменениям в классе.
- Недостатки: Значительно сложнее в реализации. Требуется вручную написать методы writeExternal и readExternal, а также обязательно создать публичный конструктор без аргументов. Легко допустить ошибку (например, перепутать порядок чтения полей).

```

TabulatedFunction.java U X
src > functions > TabulatedFunction.java > ...
1 package functions;
2
3 /* 1. Расширяем Function (extends Function)
4 | 2. Добавляем Serializable для Задания 9 */
5 public interface TabulatedFunction extends Function, java.io.Serializable {
6     /* Методы getLeftDomainBorder, getRightDomainBorder, getFunctionValue
7     теперь наследуются из Function. Удаляем их отсюда. */
8
9     int getPointsCount();
10    FunctionPoint getPoint(int index);
11    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;
12    double getPointX(int index);
13    void setPointX(int index, double x) throws InappropriateFunctionPointException;
14    double getPointY(int index);
15    void setPointY(int index, double y);
16    void deletePoint(int index);
17    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
18 }
19

FunctionPoint.java U X
src > functions > FunctionPoint.java > ...
1 package functions;
2
3 // Реализуем Serializable для Задания 9
4 public class FunctionPoint implements java.io.Serializable {
5     private double x;
6     private double y;
7
8     // Обязательный элемент для Serializable
9     private static final long serialVersionUID = 1L;
10
11     public FunctionPoint(double x, double y) {
12         this.x = x;
13         this.y = y;
14     }
15
16     public FunctionPoint(FunctionPoint point) {
17         this.x = point.x;
18         this.y = point.y;
19     }
20
21     public FunctionPoint() {
22         this.x = 0;
23         this.y = 0;
24     }
25
26     public double getX() {
27         return x;
28     }
29
30     public double getY() {
31         return y;
32     }
33
34     public void setX(double x) {
35         this.x = x;
36     }
37
38     public void setY(double y) {
39         this.y = y;
40     }
41 }
42

ArrayTabulatedFunction.java X
src > functions > ArrayTabulatedFunction.java ...
1 package functions;
2
3 import java.io.Serializable;
4
5 // Класс-последовательное TabulatedFunction, который несет в себе массив Function с Serializable
6 public class ArrayTabulatedFunction implements TabulatedFunction {
7     private static final long serialVersionUID = 1L; // Для Java 9
8
9     // Конструктор
10    public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
11        if (leftX >= rightX || pointsCount < 3) {
12            throw new IllegalArgumentException("Invalid arguments for function creation: leftX >= rightX or pointsCount < 3");
13        }
14
15        this.points = new FunctionPoint[pointsCount];
16        this.step = (rightX - leftX) / (pointsCount - 1);
17        for (int i = 0; i < pointsCount; ++i) {
18            points[i] = new FunctionPoint(leftX + i * step, 0);
19        }
20    }
21
22    // Конструктор
23    public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {
24        int count = values.length;
25        if (leftX >= rightX || count < 2) {
26            throw new IllegalArgumentException("Invalid arguments for function creation: leftX >= rightX or pointsCount < 2");
27        }
28
29        this.points = new FunctionPoint[count];
30        double step = (rightX - leftX) / (count - 1);
31        for (int i = 0; i < count; ++i) {
32            points[i] = new FunctionPoint(leftX + i * step, values[i]);
33        }
34    }
35
36    // Именной конструктор #4 (Задание 1)
37    public ArrayTabulatedFunction(FunctionPoint[] points) {
38        if (points.length < 2) {
39            throw new IllegalArgumentException("Function must have at least 2 points");
40        }
41
42        // Проверка на упорядоченность
43        for (int i = 0; i < points.length - 1; ++i) {
44            if (points[i].getX() >= points[i + 1].getX()) {
45                throw new IllegalArgumentException("Points are not strictly sorted by X");
46            }
47        }
48
49        // Проверка на одинаковые значения
50        for (int i = 0; i < points.length; ++i) {
51            this.points[i] = new FunctionPoint(points[i]);
52        }
53    }
54
55    public double getLeftDomainBorder() {
56        return points[0].getX();
57    }
58
59    public double getRightDomainBorder() {
60        return points[points.length - 1].getX();
61    }
62
63    public double getFunctionValue(double x) {
64        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
65            return Double.NaN;
66        }
67
68        for (int i = 0; i < points.length - 1; ++i) {
69            if (points[i].getX() <= x && x <= points[i + 1].getX())) {
70                double y1 = points[i].getY();

```



```

MainJava.t.m X
src > MainJava > ...
1 import functions.*;
2 import functions.basic.*;
3 import java.io.*;
4
5 public class Main {
6
7     Run|Debug
8     public static void main(String[] args) {
9
10        // --- Задание 8, п.1: Sin и Cos ---
11        System.out.println(x: "---- Задание 8.1: Sin и Cos ---");
12        Function sin = new Sin();
13        Function cos = new Cos();
14        for (double x = 0; x <= Math.PI; x += 0.1) {
15            System.out.printf(format: "x=%1.4f, sin(x)=%1.4f, cos(x)=%1.4f\n", x, sin.getFunctionValue(x), cos.getFunctionValue(x));
16        }
17
18        // --- Задание 8, п.2: Табулирование Sin и Cos ---
19        System.out.println(x: "An-- Задание 8.2: Табулирование Sin и Cos (10 точек) ---");
20        TabulatedFunction tabSin = TabulatedFunctions.tabulate(sin, leftX: 0, Math.PI, pointsCount: 10);
21        TabulatedFunction tabCos = TabulatedFunctions.tabulate(cos, leftX: 0, Math.PI, pointsCount: 10);
22        for (double x = 0; x <= Math.PI; x += 0.1) {
23            System.out.printf(format: "x=%1.4f, tabSin(x)=%1.4f, tabCos(x)=%1.4f\n", x, tabSin.getFunctionValue(x), tabCos.getFunctionValue(x));
24        }
25
26        // --- Задание 8, п.3: Сумма квадратов ---
27        System.out.println(x: "An-- Задание 8.3: Сумма квадратов (sin^2 + cos^2) ---");
28        Function sinSqr = Functions.power(tabSin, power: 2);
29        Function cosSqr = Functions.power(tabCos, power: 2);
30        Function sumSqr = Functions.sum(sinSqr, cosSqr);
31        for (double x = 0; x <= Math.PI; x += 0.1) {
32            System.out.printf(format: "x=%1.4f, sin^2+cos^2=%1.4f\n", x, sumSqr.getFunctionValue(x));
33        }
34
35        // --- Задание 8, п.4: Тест write/read (символический поток) ---
36        System.out.println(x: "An-- Задание 8.4: Тест write/read (текстовый файл) ---");
37        try {
38            TabulatedFunction tabExp = TabulatedFunctions.tabulate(new Exp(), leftX: 0, rightX: 10, pointsCount: 11);
39
40            // Запись в файл
41            Writer out = new FileWriter(fileName: "tabulated-exp.txt");
42            TabulatedFunctions.writeTabulatedFunction(tabExp, out);
43            out.close();
44
45            // Чтение из файла
46            Reader in = new FileReader(fileName: "tabulated-exp.txt");
47            TabulatedFunction readExp = TabulatedFunctions.readTabulatedFunction(in);
48            in.close();
49
50        } catch (IOException e) {
51            e.printStackTrace();
52        }
53
54    }
55
56    // --- Задание 8, п.5: Тест output/input (байтовый поток) ---
57    System.out.println(x: "An-- Задание 8.5: Тест output/input (бинарный файл) ---");
58    try {
59        TabulatedFunction tabLog = TabulatedFunctions.tabulate(new Log(Math.E), leftX: 1, rightX: 10, pointsCount: 10);
60
61        // Запись в файл
62        OutputStream out = new FileOutputStream(name: "tabulated-log.dat");
63        TabulatedFunctions.outputTabulatedFunction(tabLog, out);
64        out.close();
65
66        // Чтение из файла
67        InputStream in = new FileInputStream(name: "tabulated-log.dat");
68        TabulatedFunction readLog = TabulatedFunctions.inputTabulatedFunction(in);
69        in.close();
70
71        // Сравнение
72        System.out.println(x: "Сравнение исходной и считанной из файла функции:");
73        for (double x = 0; x <= 10; x += 1) {
74            System.out.printf(format: "x=%1.4f, Orig(x)=%1.2f, Read(x)=%1.2f\n", x, tabLog.getFunctionValue(x), readLog.getFunctionValue(x));
75        }
76
77    } catch (IOException e) {
78        e.printStackTrace();
79    }
80
81    // --- Задание 8, п.6: Тест СерIALIZации (Serializable) ---
82    System.out.println(x: "An-- Задание 9: Тест СерIALIZации (Serializable, ArrayTabulatedFunction) ---");
83    try {
84        // log(exp(x)) - x
85        Function f = Functions.composition(new Log(Math.E), new Exp());
86        // Используем ArrayTabulatedFunction для теста Serializable
87    }
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
735
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
151
```

```
src > MainJava1_M
5 public class Main {
7     public static void main(String[] args) {
8         // log(exp(x)) = x
9         Function f = Functions.composition(new Log(Math.E), new Exp());
10        // Используем TabulatedFunction для теста Serializable
11        TabulatedFunction tabFunc = TabulatedFunctions.tabulate(f, leftX: 0, rightX: 10, pointsCount: 11);
12
13        // Сериализация (запись объекта)
14        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(name: "serialized-array.ser"));
15        out.writeObject(tabFunc);
16        out.close();
17
18        // Десериализация (чтение объекта)
19        ObjectInputStream in = new ObjectInputStream(new FileInputStream(name: "serialized-array.ser"));
20        TabulatedFunction readFunc = (TabulatedFunction) in.readObject();
21        in.close();
22
23        // Сравнение
24        System.out.println(x: "Сравнение исходной и десериализованной функции (Array):");
25        for (double x = 0; x < 10; x += 1) {
26            System.out.printf(format: "%-4.1f, Orig(x)=%.2f, Read(x)=%.2f\n", x, tabFunc.getFunctionValue(x), readFunc.getFunctionValue(x));
27        }
28
29    } catch (IOException | ClassNotFoundException e) {
30        e.printStackTrace();
31    }
32
33    // --- ДОБАВЛЕННЫЙ ТЕСТ: Задание 9 (Externalizable) ---
34    System.out.println(x: "\n--- Задание 9: Тест Сериализации (Externalizable, LinkedListTabulatedFunction) ---");
35    try {
36        // Создаем функцию f(x) = x^2
37        Function sqr = new Power(new Function() {
38            public double getLeftDomainBorder() { return Double.NEGATIVE_INFINITY; }
39            public double getRightDomainBorder() { return Double.POSITIVE_INFINITY; }
40            public double getFunctionValue(double x) { return x * x; }
41        }, 2);
42
43        // Создаем табулированную функцию на основе LinkedListTabulatedFunction
44        double[] values = {0, 1, 4, 9, 16};
45        TabulatedFunction tabFuncLinked = new LinkedListTabulatedFunction(leftX: 0, rightX: 4, values);
46
47        System.out.println(x: "Исходная функция (LinkedList):");
48        for (int i = 0; i < tabFuncLinked.getPointsCount(); i++) {
49            System.out.printf(format: "Point №%d: (%.1f, %.1f)\n", i, tabFuncLinked.getPointX(i), tabFuncLinked.getPointY(i));
50        }
51
52        // Сериализация (запись объекта)
53        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(name: "serialized-linkedlist.ser"));
54
55        TabulatedFunction tabFuncUnlinked = new LinkedListTabulatedFunction(leftX: 0, rightX: 4, values);
56
57        System.out.println(x: "Исходная функция (LinkedList):");
58        for (int i = 0; i < tabFuncUnlinked.getPointsCount(); i++) {
59            System.out.printf(format: "Point №%d: (%.1f, %.1f)\n", i, tabFuncUnlinked.getPointX(i), tabFuncUnlinked.getPointY(i));
60        }
61
62        // Сериализация (запись объекта)
63        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(name: "serialized-linkedlist.ser"));
64        out.writeObject(tabFuncUnlinked);
65        out.close();
66
67        // Десериализация (чтение объекта)
68        ObjectInputStream in = new ObjectInputStream(new FileInputStream(name: "serialized-linkedlist.ser"));
69        TabulatedFunction readFuncLinked = (TabulatedFunction) in.readObject();
70        in.close();
71
72        // Сравнение
73        System.out.println(x: "Сравнение исходной и десериализованной функции (LinkedList):");
74        for (double x = 0; x < 4; x += 0.5) {
75            System.out.printf(format: "%-4.1f, Orig(x)=%.2f, Read(x)=%.2f\n", x, tabFuncLinked.getFunctionValue(x), readFuncLinked.getFunctionValue(x));
76        }
77
78    } catch (IOException | ClassNotFoundException e) {
79        e.printStackTrace();
80    }
81
82 }
83
84
85 InExceptionMessages` '-cp' 'C:\Users\Borus\AppData\Roaming\Code\User\workspaceStorage\3c1765a2ca51b4f45\ai2fef7ac29714\redhat.java\jdt_ws\Lab-4-2025_44202e57\bin' 'Main' ...
● PS C:\projects\Lab-4-2025> cd ..& cd 'c:\projects\Lab-4-2025'; & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:58520' '-XX:+ShowCodeDetailsInExceptionMessages` '-cp' 'C:\Users\Borus\AppData\Roaming\Code\User\workspaceStorage\3c1765a2ca51b4f45\ai2fef7ac29714\redhat.java\jdt_ws\Lab-4-2025_44202e57\bin' 'Main'
--- Задание 8.1: Sin и Cos ---
x=0,0, sin(x)=0,0000, cos(x)=1,0000
x=0,1, sin(x)=0,9998, cos(x)=-0,9950
x=0,2, sin(x)=0,9987, cos(x)=-0,9801
x=0,3, sin(x)=0,9955, cos(x)=-0,9553
x=0,4, sin(x)=0,9894, cos(x)=-0,9211
x=0,5, sin(x)=0,9794, cos(x)=-0,8776
x=0,6, sin(x)=0,9546, cos(x)=-0,8253
x=0,7, sin(x)=0,9142, cos(x)=-0,7648
x=0,8, sin(x)=0,7174, cos(x)=-0,6967
x=0,9, sin(x)=0,7833, cos(x)=-0,6216
x=1,0, sin(x)=0,8415, cos(x)=0,5403
x=1,1, sin(x)=0,8912, cos(x)=-0,4536
x=1,2, sin(x)=0,9320, cos(x)=-0,3624
x=1,3, sin(x)=0,9636, cos(x)=-0,2675
x=1,4, sin(x)=0,9854, cos(x)=-0,1700
x=1,5, sin(x)=0,9975, cos(x)=-0,0707
x=1,6, sin(x)=0,9996, cos(x)=-0,0292
x=1,7, sin(x)=0,9917, cos(x)=-0,1288
x=1,8, sin(x)=0,9738, cos(x)=-0,2272
x=1,9, sin(x)=0,9463, cos(x)=-0,3233
x=2,0, sin(x)=0,9093, cos(x)=-0,4161
x=2,1, sin(x)=0,8632, cos(x)=-0,5048
x=2,2, sin(x)=0,8085, cos(x)=-0,5885
x=2,3, sin(x)=0,7457, cos(x)=-0,6663
x=2,4, sin(x)=0,6755, cos(x)=-0,7374
x=2,5, sin(x)=0,5985, cos(x)=-0,8011
x=2,6, sin(x)=0,5155, cos(x)=-0,8569
x=2,7, sin(x)=0,4274, cos(x)=-0,9041
x=2,8, sin(x)=0,3350, cos(x)=-0,9422
x=2,9, sin(x)=0,2392, cos(x)=-0,9710
x=3,0, sin(x)=0,1411, cos(x)=-0,9900
x=3,1, sin(x)=0,0416, cos(x)=-0,9991
--- Задание 8.2: Табулирование Sin и Cos (10 точек) ---
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  Debug: Main +    ... |         PS C:\projects\Lab-4-2025> c:& cd 'c:\projects\Lab-4-2025'; & 'C:\Program Files\Java\jdk-24\bin\java.exe' -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=local  
x=3,1, tabsin(x)=0,0408, tabcos(x)=0,9928  
--- Задание 8.3: Сумма квадратов ( $\sin^2 x + \cos^2 x$ ) ---  
x=0,0, sin^2+cos^2=1,0000  
x=0,1, sin^2+cos^2=0,9753  
x=0,2, sin^2+cos^2=0,9705  
x=0,3, sin^2+cos^2=0,9854  
x=0,4, sin^2+cos^2=0,9850  
x=0,5, sin^2+cos^2=0,9704  
x=0,6, sin^2+cos^2=0,9756  
x=0,7, sin^2+cos^2=0,9994  
x=0,8, sin^2+cos^2=0,9751  
x=0,9, sin^2+cos^2=0,9706  
x=1,0, sin^2+cos^2=0,9859  
x=1,1, sin^2+cos^2=0,9845  
x=1,2, sin^2+cos^2=0,9703  
x=1,3, sin^2+cos^2=0,9759  
x=1,4, sin^2+cos^2=0,9987  
x=1,5, sin^2+cos^2=0,9748  
x=1,6, sin^2+cos^2=0,9707  
x=1,7, sin^2+cos^2=0,9864  
x=1,8, sin^2+cos^2=0,9841  
x=1,9, sin^2+cos^2=0,9702  
x=2,0, sin^2+cos^2=0,9762  
x=2,1, sin^2+cos^2=0,9981  
x=2,2, sin^2+cos^2=0,9745  
x=2,3, sin^2+cos^2=0,9708  
x=2,4, sin^2+cos^2=0,9869  
x=2,5, sin^2+cos^2=0,9836  
x=2,6, sin^2+cos^2=0,9702  
x=2,7, sin^2+cos^2=0,9765  
x=2,8, sin^2+cos^2=0,9975  
x=2,9, sin^2+cos^2=0,9743  
x=3,0, sin^2+cos^2=0,9709  
x=3,1, sin^2+cos^2=0,9873  
--- Задание 8.4: Тест write/read (текстовый файл) ---  
Сравнение исходной и считанной из файла функции:  
x=0,0, Orig(x)=1,00, Read(x)=1,00  
x=1,0, Orig(x)=0,72, Read(x)=2,72
```

```

PS C:\projects\Lab-4-2025> ::; cd 'c:\projects\Lab-4-2025'; & 'C:\Program Files\Java\jdk-24\bin\java.exe' -agentlib:jdwpt=transport=dt_socket,server=n,suspend=y,address=local
x=3,1, sin^2+cos^2=0,9873

--- Задание 8.4: Тест write/read (текстовый файл) ---
Сравнение исходной и считанной из файла функции:
x=0,0, Orig(x)=1,00, Read(x)=1,00
x=1,0, Orig(x)=2,72, Read(x)=2,72
x=2,0, Orig(x)=7,39, Read(x)=7,39
x=3,0, Orig(x)=20,09, Read(x)=20,09
x=4,0, Orig(x)=54,60, Read(x)=54,60
x=5,0, Orig(x)=148,41, Read(x)=148,41
x=6,0, Orig(x)=403,43, Read(x)=403,43
x=7,0, Orig(x)=1096,63, Read(x)=1096,63
x=8,0, Orig(x)=2980,96, Read(x)=2980,96
x=9,0, Orig(x)=8103,08, Read(x)=8103,08
x=10,0, Orig(x)=22026,47, Read(x)=22026,47

--- Задание 8.5: Тест output/input (бинарный файл) ---
Сравнение исходной и считанной из файла функции:
x=1,0, Orig(x)=0,00, Read(x)=0,00
x=2,0, Orig(x)=0,69, Read(x)=0,69
x=3,0, Orig(x)=1,10, Read(x)=1,10
x=4,0, Orig(x)=1,39, Read(x)=1,39
x=5,0, Orig(x)=1,61, Read(x)=1,61
x=6,0, Orig(x)=1,79, Read(x)=1,79
x=7,0, Orig(x)=1,95, Read(x)=1,95
x=8,0, Orig(x)=2,08, Read(x)=2,08
x=9,0, Orig(x)=2,20, Read(x)=2,20
x=10,0, Orig(x)=2,30, Read(x)=2,30

--- Задание 9: Тест СерIALIZации (Serializable, ArrayTabulatedFunction) ---
Сравнение исходной и десериализованной функции (Array):
x=0,0, Orig(x)=0,00, Read(x)=0,00
x=1,0, Orig(x)=1,00, Read(x)=1,00
x=2,0, Orig(x)=2,00, Read(x)=2,00
x=3,0, Orig(x)=3,00, Read(x)=3,00
x=4,0, Orig(x)=4,00, Read(x)=4,00
x=5,0, Orig(x)=5,00, Read(x)=5,00
x=6,0, Orig(x)=6,00, Read(x)=6,00
x=7,0, Orig(x)=7,00, Read(x)=7,00
x=8,0, Orig(x)=8,00, Read(x)=8,00

```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS ⚡ Debug: Main + - ⬤ ⬧ ... ×
PS C:\projects\Lab-4-2025> c;; cd 'c:\projects\Lab-4-2025'; & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost'
x=>7,0, Orig(x)=1,95, Read(x)=1,95
x=>8,0, Orig(x)=2,08, Read(x)=2,08
x=>9,0, Orig(x)=2,20, Read(x)=2,20
x=>10,0, Orig(x)=2,30, Read(x)=2,30

--- Задание 9: Тест СерIALIZации (Serializable, ArrayTabulatedFunction) ---
Сравнение исходной и десериализованной функции (Array):
x=>0,0, Orig(x)=0,00, Read(x)=0,00
x=>1,0, Orig(x)=1,00, Read(x)=1,00
x=>2,0, Orig(x)=2,00, Read(x)=2,00
x=>3,0, Orig(x)=3,00, Read(x)=3,00
x=>4,0, Orig(x)=4,00, Read(x)=4,00
x=>5,0, Orig(x)=5,00, Read(x)=5,00
x=>6,0, Orig(x)=6,00, Read(x)=6,00
x=>7,0, Orig(x)=7,00, Read(x)=7,00
x=>8,0, Orig(x)=8,00, Read(x)=8,00
x=>9,0, Orig(x)=9,00, Read(x)=9,00
x=>10,0, Orig(x)=10,00, Read(x)=10,00

--- Задание 9: Тест СерIALIZации (Externalizable, LinkedListTabulatedFunction) ---
Исходная функция (LinkedList):
Point 0: (0,0, 0,0)
Point 1: (1,0, 1,0)
Point 2: (2,0, 4,0)
Point 3: (3,0, 9,0)
Point 4: (4,0, 16,0)
Сравнение исходной и десериализованной функции (LinkedList):
x=>0,0, Orig(x)=0,00, Read(x)=0,00
x=>0,5, Orig(x)=0,50, Read(x)=0,50
x=>1,0, Orig(x)=1,00, Read(x)=1,00
x=>1,5, Orig(x)=2,50, Read(x)=2,50
x=>2,0, Orig(x)=4,00, Read(x)=4,00
x=>2,5, Orig(x)=6,50, Read(x)=6,50
x=>3,0, Orig(x)=9,00, Read(x)=9,00
x=>3,5, Orig(x)=12,50, Read(x)=12,50
x=>4,0, Orig(x)=16,00, Read(x)=16,00
PS C:\projects\Lab-4-2025>
```

