

Лабораторная работа №4

Выполнила: Аксенова Алина Владимировна

Номер группы: 6204-010302D

Дата выполнения: 2025г.

Оглавление

Задание1.....	3
Задание2.....	3
Задание3.....	4
Задание4.....	5
Задание5.....	5
Задание6.....	6
Задание7.....	7
Задание8.....	8
Задание9.....	9

Задание 1

Для реализации конструкторов, получающих массив точек, я добавила в классы `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` новые конструкторы. В конструкторах я проверяю, что количество точек не менее двух и что они упорядочены по возрастанию X с использованием машинного эпсилона $1e-10$.

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

    // Проверка упорядоченности точек по X
    for (int i = 1; i < points.length; i++) {
        if (points[i].getX() <= points[i-1].getX() + 1e-10) {
            throw new IllegalArgumentException("Точки должны быть строго
            упорядочены по возрастанию X");
        }
    }

    this.pointsCount = points.length;
    this.points = new FunctionPoint[pointsCount + 10];

    // Копируем точки с обеспечением инкапсуляции
    for (int i = 0; i < pointsCount; i++) {
        this.points[i] = new FunctionPoint(points[i]);
    }
}
```

Задание 2

Я создала интерфейс `Function` с методами для получения границ области определения и значения функции. Затем исключила эти методы из `TabulatedFunction` и сделала так, чтобы он расширял `Function`.

```

public interface Function {
    double getLeftDomainBorder();
    double getRightDomainBorder();
    double getFunctionValue(double x);
}

public interface TabulatedFunction extends Function {
    // Специфичные методы для табулированных функций
    int getPointsCount();
    FunctionPoint getPoint(int index);
    void setPoint(int index, FunctionPoint point) throws
    InappropriateFunctionPointException;
    // ... остальные методы
}

```

Задание 3

В пакете `functions.basic` я создала классы для аналитических функций. Класс `Exp` вычисляет экспоненту, класс `Log` принимает основание в конструкторе. Для тригонометрических функций создала базовый класс `TrigonometricFunction`.

```

public class Exp implements Function {
    public double getLeftDomainBorder() {
        return Double.NEGATIVE_INFINITY;
    }
    public double getRightDomainBorder() {
        return Double.POSITIVE_INFINITY;
    }
    public double getFunctionValue(double x) {
        return Math.exp(x);
    }
}

```

Задание 4

В пакете `functions.meta` я реализовала классы для комбинирования функций. Класс `Sum` представляет сумму двух функций, `Scale` масштабирует функцию вдоль осей.

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

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

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

    public double getFunctionValue(double x) {
        return f1.getFunctionValue(x) + f2.getFunctionValue(x);
    }
}
```

Задание 5

Я создала класс `Functions` с приватным конструктором и статическими методами-фабриками для создания комбинированных функций.

```
public final class Functions {
    private Functions() {
        throw new UnsupportedOperationException("Нельзя создать объект класса Functions");
    }

    public static Function shift(Function f, double shiftX, double
```

```

    shiftY) {
        return new Shift(f, shiftX, shiftY);
    }

    public static Function sum(Function f1, Function f2) {
        return new Sum(f1, f2);
    }
}

    return new Shift(f, shiftX, shiftY);
}

    public static Function scale(Function f, double scaleX, double
scaleY) {
        return new Scale(f, scaleX, scaleY);
    }
}

```

Задание 6

Я добавила в `TabulatedFunctions` МЕТОД `tabulate()`, КОТОРЫЙ создает табулированный аналог функции на заданном отрезке.

```

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

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

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

```

```

    }

    return new ArrayTabulatedFunction(points);
}

```

Задание 7

Для работы с потоками ввода-вывода я добавила в `TabulatedFunctions` четыре метода. Для байтовых потоков использую `DataOutputStream/DataInputStream`, для СИМВОЛЬНЫХ - `PrintWriter/StreamTokenizer`.

```

public static void outputTabulatedFunction(TabulatedFunction
function, OutputStream out) throws IOException {
    DataOutputStream dos = new DataOutputStream(out);
    dos.writeInt(function.getPointsCount());

    for (int i = 0; i < function.getPointsCount(); i++) {
        FunctionPoint point = function.getPoint(i);
        dos.writeDouble(point.getX());
        dos.writeDouble(point.getY());
    }

    dos.flush();
}

```

```

    public static TabulatedFunction
readTabulatedFunction(Reader in) throws IOException {
    StreamTokenizer tokenizer = new StreamTokenizer(in);
    tokenizer.parseNumbers();

    tokenizer.nextToken();
    int pointsCount = (int) tokenizer.nval;

    FunctionPoint[] points = new FunctionPoint[pointsCount];

    for (int i = 0; i < pointsCount; i++) {

```

```

        tokenizer.nextToken();
        double x = tokenizer.nval;
        tokenizer.nextToken();
        double y = tokenizer.nval;
        points[i] = new FunctionPoint(x, y);
    }

    return new ArrayTabulatedFunction(points);
}

```

Задание 8

Для комплексного тестирования я создала класс `Main` с тестовыми методами. Я тестировала базовые функции, создавала табулированные аналоги, сравнивала точность при разном количестве точек, тестировала комбинации функций и ввод-вывод в файлы.

```

Sin sin = new Sin();
Cos cos = new Cos();
TabulatedFunction tabulatedSin = TabulatedFunctions.tabulate(sin,
0, Math.PI, 10);
Function sumOfSquares = Functions.sum(
    Functions.power(tabulatedSin, 2),
    Functions.power(tabulatedCos, 2)
);

    Functions.power(tabulatedSin, 2),
    Functions.power(tabulatedCos, 2)
);

```


Задание 9

Для поддержки сериализации я реализовала интерфейс `Externalizable` в классе `ArrayTabulatedFunction`. Я добавила конструктор без аргументов и методы для ручного управления процессом сериализации.

```
public class ArrayTabulatedFunction implements TabulatedFunction,
    java.io.Serializable, java.io.Externalizable {

    // Конструктор без аргументов для Externalizable
    public ArrayTabulatedFunction() {
        this.points = new FunctionPoint[2];
        this.points[0] = new FunctionPoint(0.0, 0.0);
        this.points[1] = new FunctionPoint(1.0, 1.0);
        this.pointsCount = 2;
    }

    // Реализация Externalizable
    public void writeExternal(ObjectOutput out) throws IOException
    {
        out.writeInt(pointsCount);
        for (int i = 0; i < pointsCount; i++) {
            out.writeDouble(points[i].getX());
            out.writeDouble(points[i].getY());
        }
    }

    public void readExternal(ObjectInput in) throws IOException {
        pointsCount = in.readInt();
        points = new FunctionPoint[pointsCount];
        for (int i = 0; i < pointsCount; i++) {
            double x = in.readDouble();
            double y = in.readDouble();
            points[i] = new FunctionPoint(x, y);
        }
    }
}
```

Вывод Main:

```
=== Лабораторная работа #4 ===

--- Тестирование базовых функций ---
Exp: f(0) = 1.0
Exp: f(1) = 2.718281828459045
Ln: f(1) = 0.0
Ln: f(Math.E) = 1.0
Sin(0) = 0.0
Cos(0) = 1.0
Sin(pi/2) = 1.0

--- Тестирование комбинированных функций ---
sin?(x) + cos?(x) для различных x:
  x=0,00: 1,000000
  x=0,79: 1,000000
  x=1,57: 1,000000
  x=2,36: 1,000000
  x=3,14: 1,000000
exp(ln(5)) = 5.0

--- Тестирование табулирования и ввода/вывода ---
Sin(x) на отрезке [0, pi] с шагом 0.1:
  sin(0,0) = 0,000000
  sin(0,1) = 0,099833
  sin(0,2) = 0,198669
  sin(0,3) = 0,295520
  sin(0,4) = 0,389418
  sin(0,5) = 0,479426
  sin(0,6) = 0,564642
  sin(0,7) = 0,644218
  sin(0,8) = 0,717356
  sin(0,9) = 0,783327
  sin(1,0) = 0,841471
  sin(1,1) = 0,891207
  sin(1,2) = 0,932039
  sin(1,3) = 0,963558
  sin(1,4) = 0,985450
  sin(1,5) = 0,997495
  sin(1,6) = 0,999574
  sin(1,7) = 0,991665
  sin(1,8) = 0,973848
  sin(1,9) = 0,946300
  sin(2,0) = 0,909297
  sin(2,1) = 0,863209
```

sin(2,2) = 0,808496
sin(2,3) = 0,745705
sin(2,4) = 0,675463
sin(2,5) = 0,598472
sin(2,6) = 0,515501
sin(2,7) = 0,427380
sin(2,8) = 0,334988
sin(2,9) = 0,239249
sin(3,0) = 0,141120
sin(3,1) = 0,041581

Cos(x) на отрезке [0, pi] с шагом 0.1:

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

Табулированный синус (0 до pi, 10 точек):

(0,00, 0,0000)
(0,35, 0,3420)

(0,70, 0,6428)
(1,05, 0,8660)
(1,40, 0,9848)
(1,75, 0,9848)
(2,09, 0,8660)
(2,44, 0,6428)
(2,79, 0,3420)
(3,14, 0,0000)

Табулированный косинус (0 до π , 10 точек):

(0,00, 1,0000)
(0,35, 0,9397)
(0,70, 0,7660)
(1,05, 0,5000)
(1,40, 0,1736)
(1,75, -0,1736)
(2,09, -0,5000)
(2,44, -0,7660)
(2,79, -0,9397)
(3,14, -1,0000)

Сравнение исходного и табулированного синуса:

x=0,0: исходный=0,000000, табулированный=0,000000, разница=0,000000
x=0,1: исходный=0,099833, табулированный=0,097982, разница=0,001852
x=0,2: исходный=0,198669, табулированный=0,195963, разница=0,002706
x=0,3: исходный=0,295520, табулированный=0,293945, разница=0,001576
x=0,4: исходный=0,389418, табулированный=0,385907, разница=0,003512
x=0,5: исходный=0,479426, табулированный=0,472070, разница=0,007355
x=0,6: исходный=0,564642, табулированный=0,558234, разница=0,006409
x=0,7: исходный=0,644218, табулированный=0,643982, разница=0,000235
x=0,8: исходный=0,717356, табулированный=0,707935, разница=0,009421
x=0,9: исходный=0,783327, табулированный=0,771888, разница=0,011439
x=1,0: исходный=0,841471, табулированный=0,835841, разница=0,005630
x=1,1: исходный=0,891207, табулированный=0,883993, разница=0,007214
x=1,2: исходный=0,932039, табулированный=0,918022, разница=0,014017
x=1,3: исходный=0,963558, табулированный=0,952051, разница=0,011508
x=1,4: исходный=0,985450, табулированный=0,984808, разница=0,000642
x=1,5: исходный=0,997495, табулированный=0,984808, разница=0,012687
x=1,6: исходный=0,999574, табулированный=0,984808, разница=0,014766
x=1,7: исходный=0,991665, табулированный=0,984808, разница=0,006857
x=1,8: исходный=0,973848, табулированный=0,966204, разница=0,007644
x=1,9: исходный=0,946300, табулированный=0,932175, разница=0,014125
x=2,0: исходный=0,909297, табулированный=0,898147, разница=0,011151
x=2,1: исходный=0,863209, табулированный=0,862441, разница=0,000768
x=2,2: исходный=0,808496, табулированный=0,798488, разница=0,010008
x=2,3: исходный=0,745705, табулированный=0,734535, разница=0,011170
x=2,4: исходный=0,675463, табулированный=0,670582, разница=0,004881
x=2,5: исходный=0,598472, табулированный=0,594072, разница=0,004401

x=2,6: исходный=0,515501, табулированный=0,507908, разница=0,007593
x=2,7: исходный=0,427380, табулированный=0,421745, разница=0,005635
x=2,8: исходный=0,334988, табулированный=0,334698, разница=0,000290
x=2,9: исходный=0,239249, табулированный=0,236716, разница=0,002533
x=3,0: исходный=0,141120, табулированный=0,138735, разница=0,002385
x=3,1: исходный=0,041581, табулированный=0,040753, разница=0,000828

$\sin(x) + \cos(x)$ через табулированные функции:

x=0,00: 1,000000
x=0,10: 0,975345
x=0,20: 0,970488
x=0,30: 0,985429
x=0,40: 0,984968
x=0,50: 0,970398
x=0,60: 0,975624
x=0,70: 0,999358
x=0,80: 0,975073
x=0,90: 0,970586
x=1,00: 0,985897
x=1,10: 0,984515
x=1,20: 0,970314
x=1,30: 0,975910
x=1,40: 0,998723
x=1,50: 0,974808
x=1,60: 0,970691
x=1,70: 0,986371
x=1,80: 0,984068
x=1,90: 0,970237
x=2,00: 0,976203
x=2,10: 0,998094
x=2,20: 0,974549
x=2,30: 0,970802
x=2,40: 0,986852
x=2,50: 0,983628
x=2,60: 0,970167
x=2,70: 0,976503
x=2,80: 0,997473
x=2,90: 0,974298
x=3,00: 0,970920
x=3,10: 0,987341

Исследование влияния количества точек на точность:

5 точек: погрешность в $\pi/2 = 0,00000000$
10 точек: погрешность в $\pi/2 = 0,01519225$
20 точек: погрешность в $\pi/2 = 0,00341551$
50 точек: погрешность в $\pi/2 = 0,00051378$

--- Тестирование ввода/вывода в файлы ---

Сравнение исходной и прочитанной экспоненты (символьные потоки):

x=0,0: исходная=1,0000, прочитанная=1,0000
x=1,0: исходная=2,7183, прочитанная=2,7183
x=2,0: исходная=7,3891, прочитанная=7,3891
x=3,0: исходная=20,0855, прочитанная=20,0855
x=4,0: исходная=54,5982, прочитанная=54,5982
x=5,0: исходная=148,4132, прочитанная=148,4132
x=6,0: исходная=403,4288, прочитанная=403,4288
x=7,0: исходная=1096,6332, прочитанная=1096,6332
x=8,0: исходная=2980,9580, прочитанная=2980,9580
x=9,0: исходная=8103,0839, прочитанная=8103,0839
x=10,0: исходная=22026,4658, прочитанная=22026,4658

Сравнение исходного и прочитанного логарифма (байтовые потоки):

x=1,0: исходная=0,0000, прочитанная=0,0000
x=2,0: исходная=0,6849, прочитанная=0,6849
x=3,0: исходная=1,0916, прочитанная=1,0916
x=4,0: исходная=1,3809, прочитанная=1,3809
x=5,0: исходная=1,6055, прочитанная=1,6055
x=6,0: исходная=1,7889, прочитанная=1,7889
x=7,0: исходная=1,9440, прочитанная=1,9440
x=8,0: исходная=2,0783, прочитанная=2,0783
x=9,0: исходная=2,1967, прочитанная=2,1967
x=10,0: исходная=2,3026, прочитанная=2,3026

--- Анализ файлов ---

Размеры файлов:

exp_text.txt (символьный): 235 байт
ln_binary.dat (байтовый): 180 байт
function_serializable.dat (сериализация): 236 байт

Преимущества/недостатки форматов:

Символьный: читаем для человека, но больший размер

Байтовый: компактный, но нечитаем для человека

Сериализация: сохраняет всю структуру объекта, но зависит от версии Java

--- Тестирование сериализации ---

Исходная функция (ln(exp(x))):

f(0,0) = 0,0000
f(1,0) = 1,0000
f(2,0) = 2,0000
f(3,0) = 3,0000
f(4,0) = 4,0000
f(5,0) = 5,0000
f(6,0) = 6,0000
f(7,0) = 7,0000
f(8,0) = 8,0000

```
f(9,0) = 9,0000  
f(10,0) = 10,0000
```

Функция сериализована в function_serializable.dat

Функция десериализована из function_serializable.dat

Сравнение исходной и десериализованной функции:

```
x=0,0: исходная=0,0000, десериализованная=0,0000  
x=1,0: исходная=1,0000, десериализованная=1,0000  
x=2,0: исходная=2,0000, десериализованная=2,0000  
x=3,0: исходная=3,0000, десериализованная=3,0000  
x=4,0: исходная=4,0000, десериализованная=4,0000  
x=5,0: исходная=5,0000, десериализованная=5,0000  
x=6,0: исходная=6,0000, десериализованная=6,0000  
x=7,0: исходная=7,0000, десериализованная=7,0000  
x=8,0: исходная=8,0000, десериализованная=8,0000  
x=9,0: исходная=9,0000, десериализованная=9,0000  
x=10,0: исходная=10,0000, десериализованная=10,0000
```

--- Тестирование Externalizable ---

Serializable размер: 108 байт

Externalizable размер: 58 байт

Разница: 50 байт