

Лабораторная работа №4

Заболотнов Николай Михайлович

6204-010302D

Task 1

В классы ArrayTabulatedFunction и LinkedListTabulatedFunction добавим конструкторы, принимающие FunctionPoint[] (рис 1 и 2).

```
public ArrayTabulatedFunction(FunctionPoint[] points) {
    if (points == null)
        throw new IllegalArgumentException("Точек меньше двух");
    if (points.length < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    for (int i = 1; i < points.length; ++i) {
        if (points[i - 1].getX() >= points[i].getX())
            throw new IllegalArgumentException("Точки не упорядоченные");
    }
    this.points = new FunctionPoint[points.length];
    for (int i = 0; i < points.length; ++i)
        this.points[i] = new FunctionPoint(points[i]);
    PointsCount = points.length;
}
```

Рис 1

```
public LinkedListTabulatedFunction(FunctionPoint[] points) {
    if (points == null)
        throw new IllegalArgumentException("Точек меньше двух");
    if (points.length < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    for (int i = 1; i < points.length; ++i) {
        if (points[i - 1].getX() >= points[i].getX())
            throw new IllegalArgumentException("Точки не упорядоченные");
    }
    head = new FunctionNode(points[0]);
    head.next = head;
    head.prev = head;
    ++size;
    for (int i = 1; i < points.length; ++i)
        addNodeToTail(new FunctionNode(points[i]));
    lastNode = head;
}
```

Рис 2

Обеспечим выброс возможных исключений. Задание выполнено.

Task 2

Создадим интерфейс Function с методами getLeftDomainBorder(), getRightDomainBorder(), getFunctionValue() (рис 3).

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

Интерфейс TabulatedFunction будет нас наследоваться от этого интерфейса. Задание выполнено.

Task 3

Создадим пакет functions.basic с некоторыми функциями (рис 4). Для тригонометрических функций создадим класс TrigonometricFunction (рис 5), от которого их наследуем. Этот класс будет абстрактным, так как не будет создаваться объект этого класса. Для вычисления значений функции будет и использовать методы из класса Math, а для обозначения области определения поля NEGATIVE_INFINITY и POSITIVE_INFINITY класса Double.

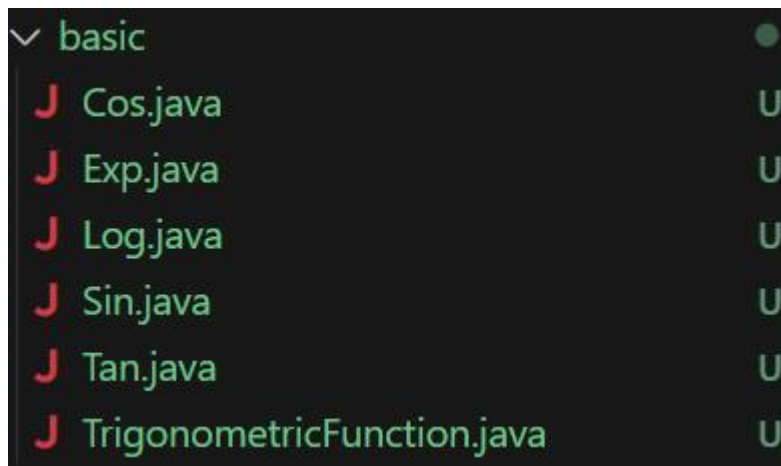


Рис 4

```
public abstract class TrigonometricFunction implements Function
```

Рис 5

Задание выполнено.

Task 4

Создадим пакет `functions.meta`, описывающий комбинированные функции (рис 6). Классы из этого пакета будут наследоваться от интерфейса `Function` следовательно наследовать методы `getLeftDomainBorder()`, `getRightDomainBorder()`, `getFunctionValue()` (например класс `Sum` на рис 7).

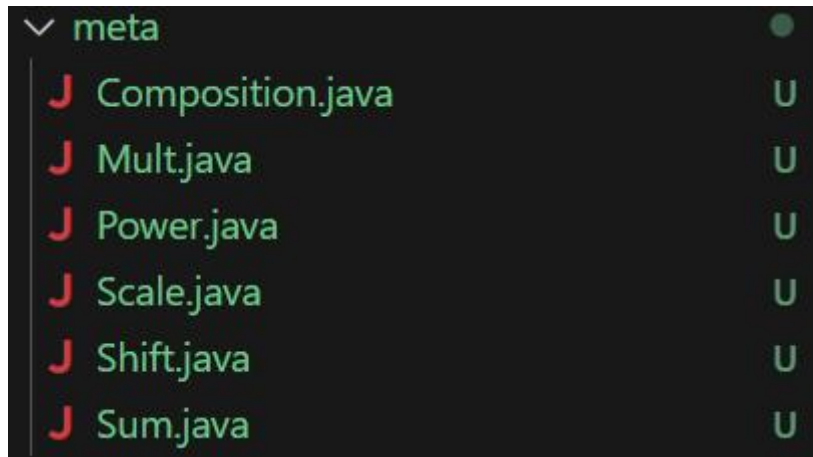


Рис 6

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

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

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

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

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

Рис 7

Задание выполнено.

Task 5

Создадим класс Functions, который содержит вспомогательные методы для создания сложных функций. Напишем статические методы shift(), scale(), power(), sum(), mult(), composition(). (рис 8).

```
public static Function shift(Function f, double shiftX, double shiftY) {  
    return new Shift(f, shiftX, shiftY);  
}  
  
public static Function scale(Function f, double scaleX, double scaleY) {  
    return new Scale(f, scaleX, scaleY);  
}  
  
public static Function power(Function f, double n) {  
    return new Power(f, n);  
}  
  
public static Function sum(Function f1, Function f2) {  
    return new Sum(f1, f2);  
}  
  
public static Function mult(Function f1, Function f2) {  
    return new Mult(f1, f2);  
}  
  
public static Function composition(Function f1, Function f2) {  
    return new Composition(f1, f2);  
}
```

Рис 8

Задание выполнено.

Task 6

Создадим класс TabulatedFunctions, в котором напишем статический метод tabulate(), принимающий функцию и возвращающий табулированный аналог этой функции на заданном отрезке с заданным количеством точек (рис 9). Обработаем выброс исключения с случае если точки выходят за область определения.


```

public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
    if (leftX >= rightX)
        throw new IllegalArgumentException("Левая граница юлбше или равна правой границе");
    if (pointsCount < 2)
        throw new IllegalArgumentException("Точек меньше двух");
    if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder())
        throw new IllegalArgumentException("Границы выходят за область отределения функции");
    FunctionPoint[] points = new FunctionPoint[pointsCount];
    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; ++i, leftX += step)
        points[i] = new FunctionPoint(leftX, function.getFunctionValue(leftX));
    return new LinkedListTabulatedFunction(points);
}

```

Рис 9

Задание выполнено.

Task 7

В этот же класс добавим статические методы `outputTabulatedFunction()`, `inputTabulatedFunction()`, `writeTabulatedFunction()`, `readTabulatedFunction()` для вывода табулированной функции в файл. Для реализации этих методов в первых трех случаях воспользуемся потоками `DataOutputStream`, `DataInputStream`, `PrintWriter` и классом `StreamTokenizer` (рис 10).

```

public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException {
    DataOutputStream dataOut = new DataOutputStream(out);
    dataOut.writeInt(function.getPointsCount());
    for (int i = 0; i < function.getPointsCount(); ++i) {
        dataOut.writeDouble(function.getPointX(i));
        dataOut.writeDouble(function.getPointY(i));
    }
    dataOut.flush();
}

public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException {
    DataInputStream dataIn = new DataInputStream(in);
    int pointsCount = dataIn.readInt();
    FunctionPoint[] points = new FunctionPoint[pointsCount];
    for (int i = 0; i < pointsCount; ++i)
        points[i] = new FunctionPoint(dataIn.readDouble(), dataIn.readDouble());
    return new LinkedListTabulatedFunction(points);
}

public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException {
    PrintWriter writer = new PrintWriter(out);
    writer.print(function.getPointsCount());
    for (int i = 0; i < function.getPointsCount(); ++i) {
        writer.print(" " + function.getPointX(i) + " " + function.getPointY(i));
    }
    writer.flush();
}

public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException {
    StreamTokenizer tokenizer = new StreamTokenizer(in);
    if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER)
        throw new IOException();
    int pointsCount = (int) tokenizer.nval();
    FunctionPoint[] points = new FunctionPoint[pointsCount];
    for (int i = 0; i < pointsCount; ++i) {
        if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER)
            throw new IOException();
        double x = tokenizer.nval();
        if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER)
            throw new IOException();
        double y = tokenizer.nval();
        points[i] = new FunctionPoint(x, y);
    }
    return new LinkedListTabulatedFunction(points);
}

```

Задание выполнено.

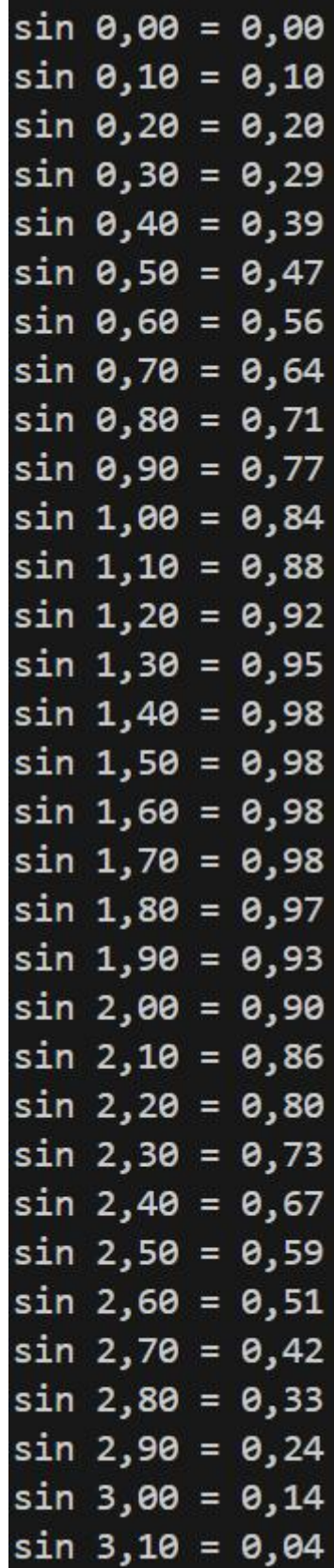
Task 8

В классе Main проверим все написанные классы и методы.

Вывод мейна:

```
00,1 = 00,0 200
80,0 = 01,0 200
70,0 = 02,0 200
20,0 = 03,0 200
10,0 = 04,0 200
08,0 = 02,0 200
18,0 = 03,0 200
07,0 = 07,0 200
00,0 = 08,0 200
10,0 = 00,0 200
42,0 = 00,1 200
24,0 = 01,1 200
03,0 = 02,1 200
02,0 = 03,1 200
71,0 = 04,1 200
70,0 = 02,1 200
00,0- = 00,1 200
01,0- = 07,1 200
22,0- = 08,1 200
23,0- = 00,1 200
14,0- = 00,2 200
02,0- = 01,2 200
82,0- = 02,2 200
00,0- = 03,2 200
07,0- = 04,2 200
07,0- = 02,2 200
48,0- = 00,2 200
08,0- = 07,2 200
40,0- = 08,2 200
00,0- = 00,2 200
80,0- = 00,3 200
00,0- = 01,3 200
```

Рис 11

A vertical black bar with white text listing sine values for angles from 0.00 to 3.10 in increments of 0.10. The text is in a monospaced font.

$\sin 0,00$	$= 0,00$
$\sin 0,10$	$= 0,10$
$\sin 0,20$	$= 0,20$
$\sin 0,30$	$= 0,29$
$\sin 0,40$	$= 0,39$
$\sin 0,50$	$= 0,47$
$\sin 0,60$	$= 0,56$
$\sin 0,70$	$= 0,64$
$\sin 0,80$	$= 0,71$
$\sin 0,90$	$= 0,77$
$\sin 1,00$	$= 0,84$
$\sin 1,10$	$= 0,88$
$\sin 1,20$	$= 0,92$
$\sin 1,30$	$= 0,95$
$\sin 1,40$	$= 0,98$
$\sin 1,50$	$= 0,98$
$\sin 1,60$	$= 0,98$
$\sin 1,70$	$= 0,98$
$\sin 1,80$	$= 0,97$
$\sin 1,90$	$= 0,93$
$\sin 2,00$	$= 0,90$
$\sin 2,10$	$= 0,86$
$\sin 2,20$	$= 0,80$
$\sin 2,30$	$= 0,73$
$\sin 2,40$	$= 0,67$
$\sin 2,50$	$= 0,59$
$\sin 2,60$	$= 0,51$
$\sin 2,70$	$= 0,42$
$\sin 2,80$	$= 0,33$
$\sin 2,90$	$= 0,24$
$\sin 3,00$	$= 0,14$
$\sin 3,10$	$= 0,04$

Рис 12


```
sin^2 0,00 + cos^2 0,00 = 1,00
sin^2 0,10 + cos^2 0,10 = 0,98
sin^2 0,20 + cos^2 0,20 = 0,97
sin^2 0,30 + cos^2 0,30 = 0,99
sin^2 0,40 + cos^2 0,40 = 0,98
sin^2 0,50 + cos^2 0,50 = 0,97
sin^2 0,60 + cos^2 0,60 = 0,98
sin^2 0,70 + cos^2 0,70 = 1,00
sin^2 0,80 + cos^2 0,80 = 0,98
sin^2 0,90 + cos^2 0,90 = 0,97
sin^2 1,00 + cos^2 1,00 = 0,99
sin^2 1,10 + cos^2 1,10 = 0,98
sin^2 1,20 + cos^2 1,20 = 0,97
sin^2 1,30 + cos^2 1,30 = 0,98
sin^2 1,40 + cos^2 1,40 = 1,00
sin^2 1,50 + cos^2 1,50 = 0,97
sin^2 1,60 + cos^2 1,60 = 0,97
sin^2 1,70 + cos^2 1,70 = 0,99
sin^2 1,80 + cos^2 1,80 = 0,98
sin^2 1,90 + cos^2 1,90 = 0,97
sin^2 2,00 + cos^2 2,00 = 0,98
sin^2 2,10 + cos^2 2,10 = 1,00
sin^2 2,20 + cos^2 2,20 = 0,97
sin^2 2,30 + cos^2 2,30 = 0,97
sin^2 2,40 + cos^2 2,40 = 0,99
sin^2 2,50 + cos^2 2,50 = 0,98
sin^2 2,60 + cos^2 2,60 = 0,97
sin^2 2,70 + cos^2 2,70 = 0,98
sin^2 2,80 + cos^2 2,80 = 1,00
sin^2 2,90 + cos^2 2,90 = 0,97
sin^2 3,00 + cos^2 3,00 = 0,97
sin^2 3,10 + cos^2 3,10 = 0,99
```

Рис 13

x = 0:	1,00	1,00
x = 1:	2,72	2,72
x = 2:	7,39	7,39
x = 3:	20,09	20,09
x = 4:	54,60	54,60
x = 5:	148,41	148,41
x = 6:	403,43	403,43
x = 7:	1096,63	1096,63
x = 8:	2980,96	2980,96
x = 9:	8103,08	8103,08
x = 10:	22026,47	22026,47
x = 0:	NaN	NaN
x = 1:	NaN	NaN
x = 2:	0,69	0,69
x = 3:	1,10	1,10
x = 4:	1,39	1,39
x = 5:	1,61	1,61
x = 6:	1,79	1,79
x = 7:	1,95	1,95
x = 8:	2,08	2,08
x = 9:	2,20	2,20
x = 10:	2,30	2,30

Рис 14

Task 9

Сделаем так, чтобы объекты всех классов, реализующих интерфейс `TabulatedFunction`, были сериализуемыми.

1. Serializable

Для этого случая достаточно добавить к сериализуемым классам интерфейс `Serializable` (рис 15).

```
public interface TabulatedFunction extends Function, Serializable
```

Рис 15

2. Externalizable

Добавим к сериализуемым классам интерфейс Externalizable (рис 16), конструкторы по умолчанию и специальные методы для ArrayTabulatedFunction (рис 17), LinkedListTabulatedFunction (рис 18) и FunctionPoint (рис 19).

```
public interface TabulatedFunction extends Function, Externalizable
```

Рис 16

```
public void writeExternal(ObjectOutput out) throws IOException {
    out.writeInt(PointsCount);
    for (int i = 0; i < PointsCount; ++i) {
        out.writeObject(points[i]);
    }
}

public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
    PointsCount = in.readInt();
    points = new FunctionPoint[PointsCount];
    for (int i = 0; i < PointsCount; ++i) {
        points[i] = (FunctionPoint) in.readObject();
    }
}
```

Рис 17

```
public void writeExternal(ObjectOutput out) throws IOException {
    out.writeInt(size);
    FunctionNode node = head;
    for (int i = 0; i < size; ++i, node = node.next) {
        out.writeObject(node.point);
    }
}

public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
    int pointsCount = in.readInt();
    head = new FunctionNode((FunctionPoint) in.readObject());
    head.next = head.prev = head;
    for (int i = 1; i < pointsCount; ++i)
        addNodeToTail(new FunctionNode((FunctionPoint) in.readObject()));
    lastNode = head;
}
```

Рис 18

```

public void writeExternal(ObjectOutput out) throws IOException {
    out.writeDouble(x);
    out.writeDouble(y);
}

public void readExternal(ObjectInput in) throws IOException {
    x = in.readDouble();
    y = in.readDouble();
}

```

Рис 19

Проверка:

x = 0:	0,00	0,00
x = 1:	1,00	1,00
x = 2:	2,00	2,00
x = 3:	3,00	3,00
x = 4:	4,00	4,00
x = 5:	5,00	5,00
x = 6:	6,00	6,00
x = 7:	7,00	7,00
x = 8:	8,00	8,00
x = 9:	9,00	9,00
x = 10:	10,00	10,00

Рис 20