

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ОТЧЕТ ПО  
ЛАБОРАТОРНОЙ РАБОТЕ № 4

**«Создание классов математических функций,  
ввод-вывод, сериализация»**

по курсу  
Объектно-ориентированное программирование

Выполнила: Гонтарь Анастасия Вячеславовна,  
студент группы 6203-010302D

## Содержание

<u>Задание №1</u> .....	3
<u>Задание №2</u> .....	3
<u>Задание №3</u> .....	4
<u>Задание №4</u> .....	6
<u>Задание №5</u> .....	9
<u>Задание №6</u> .....	9
<u>Задание №7</u> .....	10
<u>Задание №8</u> .....	12
<u>Задание №9</u> .....	16

## Задание №1

Я создала конструкторы для заданных классов, которые получают сразу все точки функции в виде массива, и обработала нужные исключения

```
public ArrayTabulatedFunction(FunctionPoint[] arrPoints) throws IllegalArgumentException { 4 usages

    if (arrPoints.length < 2)
        throw new IllegalArgumentException("Количество точек меньше двух");

    this.pointsCount = arrPoints.length;

    for (int i = 1; i < pointsCount; i++) {
        if (arrPoints[i].getX() <= arrPoints[i - 1].getX())
            throw new IllegalArgumentException("Нарушена упорядоченность");
    }

    // создаем массив точек
    this.points = new FunctionPoint[pointsCount];
    for (int i = 0; i < pointsCount; i++)
        this.points[i] = new FunctionPoint(arrPoints[i]);

}
```

  

```
public LinkedListTabulatedFunction(FunctionPoint[] arrPoints) throws IllegalArgumentException { 2 usages

    if (arrPoints.length < 2)
        throw new IllegalArgumentException("Количество точек меньше двух");

    // инициализация
    head.setPrev(head);
    head.setNext(head);
    this.pointsCount = 0;

    for (int i = 1; i < arrPoints.length; i++) {
        if (arrPoints[i].getX() <= arrPoints[i - 1].getX())
            throw new IllegalArgumentException("Нарушена упорядоченность");
    }
    // создаем список добавляя последовательно точки в конец
    for (FunctionPoint arrPoint : arrPoints)
        addNodeToTail(new FunctionPoint(arrPoint));

}
```

## Задание №2

Для выполнения этого задания я создала интерфейс в нужном пакете и описала методы

```
package functions;

public interface Function{

    public double getLeftDomainBorder(); 11 implementations

    public double getRightDomainBorder(); 11 implementations

    public double getFunctionValue(double x); 14 implementations
}
```

после исключила повторяющиеся методы из интерфейса TabulatedFunction и при помощи extends Function он стал расширять интерфейс Function.

### Задание №3

Для выполнения этого задания я создала пакет basic и описала ряд функций, заданных аналитически

#### Значение экспоненты

```
package functions.basic;

import functions.Function;

public class Exp implements Function { 2 usages

    public double getLeftDomainBorder(){
        return Double.NEGATIVE_INFINITY;
    }

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

    public double getFunctionValue(double x){
        return Math.exp(x);
    }
}
```

## Значение логарифма

```
package functions.basic;

import functions.Function;

public class Log implements Function{ 2 usages
    private double base; 2 usages

    public Log (double b){ 4 usages
        if (b <= 0 || b == 1)
            throw new IllegalArgumentException("некорректное значение для основания");
        this.base = b;
    }

    public double getLeftDomainBorder(){
        return 0;
    }
    public double getRightDomainBorder(){
        return Double.POSITIVE_INFINITY;
    }

    public double getFunctionValue(double x) {
        if (x <= 0)
            return Double.NaN;
        return Math.log(x) / Math.log(base);
    }
}
```

Для описания классов для тригонометрических функций я сначала создала интерфейс TrigonometricFunction

```
package functions.basic;

import functions.Function;

public abstract class TrigonometricFunction implements Function { 3 usages 3 inheritors

    public double getLeftDomainBorder() {
        return Double.NEGATIVE_INFINITY;
    }

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

    public abstract double getFunctionValue(double x); 3 implementations
}
```

sin

```
package functions.basic;

public class Sin extends TrigonometricFunction{ 2 usages
    public double getFunctionValue(double x){
        return Math.sin(x);
    }
}
```

COS

```
package functions.basic;

public class Cos extends TrigonometricFunction{ 2 usages
    public double getFunctionValue(double x){
        return Math.cos(x);
    }
}
```

tan

```
package functions.basic;

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

## Задание №4

Для выполнения этого задания я создала пакет meta, в котором описаны классы функций для комбинации

Сумма

```
package functions.meta;

import functions.Function;

public class Sum implements Function { 1 usage
    private Function firstFunction; 4 usages
    private Function secondFunction; 4 usages

    public Sum (Function firstFunc, Function secondFunc){ 3 usages
        if(firstFunc == null || secondFunc == null)
            throw new IllegalArgumentException("функции null");
        this.firstFunction = firstFunc;
        this.secondFunction = secondFunc;
    }

    public double getLeftDomainBorder() {
        return Math.max(firstFunction.getLeftDomainBorder(), secondFunction.getLeftDomainBorder());
    }

    public double getRightDomainBorder() {
        return Math.max(firstFunction.getRightDomainBorder(), secondFunction.getRightDomainBorder());
    }

    public double getFunctionValue(double x) {
        return firstFunction.getFunctionValue(x)+secondFunction.getFunctionValue(x);
    }
}
```

## Произведение

```
package functions.meta;

import functions.Function;

public class Mult implements Function { 1 usage
    private Function firstFunction; 4 usages
    private Function secondFunction; 4 usages

    public Mult (Function firstFunc, Function secondFunc){ 3 usages
        if(firstFunc == null || secondFunc == null)
            throw new IllegalArgumentException("функции null");
        this.firstFunction = firstFunc;
        this.secondFunction = secondFunc;
    }

    public double getLeftDomainBorder() {
        return Math.max(firstFunction.getLeftDomainBorder(), secondFunction.getLeftDomainBorder());
    }

    public double getRightDomainBorder() {
        return Math.max(firstFunction.getRightDomainBorder(), secondFunction.getRightDomainBorder());
    }

    // проверку
    public double getFunctionValue(double x) {
        return firstFunction.getFunctionValue(x)*secondFunction.getFunctionValue(x);
    }
}
```

## Возведение в степень

```
package functions.meta;

import functions.Function;

public class Power implements Function { 1 usage
    private Function function; 4 usages
    private double power; 2 usages

    public Power (Function func, double pow){ 3 usages
        if(func == null )
            throw new IllegalArgumentException("функция null");
        this.power = pow;
        this.function = func;
    }

    public double getLeftDomainBorder() {
        return function.getLeftDomainBorder();
    }

    public double getRightDomainBorder() { return function.getRightDomainBorder(); }

    public double getFunctionValue(double x){
        return Math.pow( function.getFunctionValue(x), power);
    }
}
```

## Масштабирование

```
package functions.meta;

import functions.Function;

public class Scale implements Function{ 1 usage
    private Function function; 4 usages
    private double scaleX; 3 usages
    private double scaleY; 3 usages

    public Scale(Function func, double scaleX, double scaleY){ 3 usages
        if(func == null )
            throw new IllegalArgumentException("функция null");
        this.function = func;
        this.scaleX = scaleX;
        this.scaleY = scaleY;
    }

    public double getLeftDomainBorder() {
        return function.getLeftDomainBorder()*scaleX;
    }

    public double getRightDomainBorder() {
        return function.getRightDomainBorder()*scaleY;
    }

    public double getFunctionValue(double x) {
        return function.getFunctionValue( x*x*scaleX)*scaleY;
    }
}
```

## Сдвиг по осям координат

```
package functions.meta;

import functions.Function;

public class Shift implements Function { 1 usage
    private Function function; 4 usages
    private double shiftX; 4 usages
    private double shiftY; 2 usages

    public Shift(Function func, double shiftX, double shiftY) throws IllegalArgumentException{ 3 usages
        if(func == null )
            throw new IllegalArgumentException("функция null");
        this.function = func;
        this.shiftX = shiftX;
        this.shiftY = shiftY;
    }

    public double getLeftDomainBorder(){
        return function.getLeftDomainBorder() + shiftX;
    }

    public double getRightDomainBorder(){
        return function.getRightDomainBorder() + shiftX;
    }

    public double getFunctionValue(double x){
        return function.getFunctionValue( x+ shiftX)+ shiftY;
    }
}
```

## Композиция

```
package functions.meta;

import functions.Function;

public class Composition implements Function { 1 usage
    private Function inFunction; 4 usages
    private Function outFunction; 2 usages

    public Composition(Function inFunc, Function outFunc){ 3 usages
        if(inFunc == null || outFunc == null)
            throw new IllegalArgumentException("функции null");

        this.inFunction=inFunc;
        this.outFunction=outFunc;
    }

    public double getLeftDomainBorder() {
        return inFunction.getLeftDomainBorder();
    }

    public double getRightDomainBorder() {
        return inFunction.getRightDomainBorder();
    }

    public double getFunctionValue(double x) {
        return outFunction.getFunctionValue(inFunction.getFunctionValue(x));
    }
}
```

## Задание №5

Для выполнения этого задания я создала класс Functions, для написания методов которого я использовала ранее описанные методы в пакете meta

```
package functions;

import functions.meta.*;

public class Functions { 7 usages new *
    private Functions() {} no usages new *

    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 power){ 4 usages new *
        return new Power(f, power);
    }

    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); }
}
```

## Задание №6

Для выполнения этого задания я создала класс TabulatedFunctions и описала метод, который получает функцию и возвращает ее табулированный аналог. описав конструктор и сделав его приватным, я сделала невозможным создать объект этого класса вне его

```
import java.io.*;

public class TabulatedFunctions { 11 usages
    private TabulatedFunctions() { no usages
    }

    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) { 7 usages
        if (function == null)
            throw new IllegalArgumentException("функция null");
        if (pointsCount < 2)
            throw new IllegalArgumentException("недостаточное количество точек");
    }
    if (leftX >= rightX) {
        throw new IllegalArgumentException("левая граница меньше правой");
    }
    if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
        throw new IllegalArgumentException("границы табулирования выходят за область определения");
    }

    // создаем массив для значений у
    double[] values = new double[pointsCount];
    double step = Math.abs(rightX - leftX) / (pointsCount - 1);

    // записываем значения в массив
    for (int i = 0; i < pointsCount; i++)
        values[i] = function.getFunctionValue( x leftX + i * step);

    return new ArrayTabulatedFunction(leftX, rightX, values);
}
```

## Задание №7

Для выполнения этого задания я описала методы для работы с байтовыми и символьными потоками в классе из задания 6.

Исключения пробрасываются, тем самым компилятор сам определит тип ошибки.

Потоки не закрываю, так как они могут использоваться в дальнейшем

```

// работа с байтовым потоком
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException { 1 usage
    int pointsCount = function.getPointsCount();
    DataOutputStream outputData = new DataOutputStream(out);
    outputData.writeInt(pointsCount);
    for (int i = 0; i < pointsCount; i++) {
        outputData.writeDouble(function.getPointX(i));
        outputData.writeDouble(function.getPointY(i));
    }
    outputData.flush();
}

public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException { 1 usage
    DataInputStream inputData = new DataInputStream(in);
    int pointsCount = inputData.readInt();
    FunctionPoint[] points = new FunctionPoint[pointsCount];

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

    return new ArrayTabulatedFunction(points);
}

// работа с символьным потоком
public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException{ 1 usage
    PrintWriter writer = new PrintWriter(out);
    int pointsCount = function.getPointsCount();
    writer.print(pointsCount);
    writer.print(" ");

    for (int i = 0; i < pointsCount; i++) {
        writer.print(function.getPointX(i));
        writer.print(" ");
        writer.print(function.getPointY(i));
        writer.print(" ");
    }
    writer.flush();
}

public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException { 1 usage
    StreamTokenizer tokenizer = new StreamTokenizer(in);
    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

В этом задании я проверила работу написанных классов

Вывод значений синуса и косинуса

```
"C:\Program Files\Java\jdk-24\bin\  
== Тестирование Sin и Cos ==  
Sin:  
Sin(0,00) = 0,0000  
Sin(0,10) = 0,0998  
Sin(0,20) = 0,1987  
Sin(0,30) = 0,2955  
Sin(0,40) = 0,3894  
Sin(0,50) = 0,4794  
Sin(0,60) = 0,5646  
Sin(0,70) = 0,6442  
Sin(0,80) = 0,7174  
Sin(0,90) = 0,7833  
Sin(1,00) = 0,8415  
Sin(1,10) = 0,8912  
Sin(1,20) = 0,9320  
Sin(1,30) = 0,9636  
Sin(1,40) = 0,9854  
Sin(1,50) = 0,9975  
Sin(1,60) = 0,9996  
Sin(1,70) = 0,9917  
Sin(1,80) = 0,9738  
Sin(1,90) = 0,9463  
Sin(2,00) = 0,9093  
Sin(2,10) = 0,8632  
Sin(2,20) = 0,8085
```

Cos:
$\text{Cos}(0,00) = 1,0000$
$\text{Cos}(0,10) = 0,9950$
$\text{Cos}(0,20) = 0,9801$
$\text{Cos}(0,30) = 0,9553$
$\text{Cos}(0,40) = 0,9211$
$\text{Cos}(0,50) = 0,8776$
$\text{Cos}(0,60) = 0,8253$
$\text{Cos}(0,70) = 0,7648$
$\text{Cos}(0,80) = 0,6967$
$\text{Cos}(0,90) = 0,6216$
$\text{Cos}(1,00) = 0,5403$
$\text{Cos}(1,10) = 0,4536$
$\text{Cos}(1,20) = 0,3624$
$\text{Cos}(1,30) = 0,2675$
$\text{Cos}(1,40) = 0,1700$
$\text{Cos}(1,50) = 0,0707$
$\text{Cos}(1,60) = -0,0292$
$\text{Cos}(1,70) = -0,1288$
$\text{Cos}(1,80) = -0,2272$
$\text{Cos}(1,90) = -0,3233$
$\text{Cos}(2,00) = -0,4161$
$\text{Cos}(2,10) = -0,5048$
$\text{Cos}(2,20) = -0,5885$
$\text{Cos}(2,30) = -0,6663$
$\text{Cos}(2,40) = -0,7374$

Вывод табулированных аналогов синуса и косинуса и сравнение их с исходными

```

== Тестирование и сравнение табулированных аналогов Sin и Cos ==
x = 0,00: исходный Sin = 0,0000, таб Sin 0,0000
x = 0,10: исходный Sin = 0,0998, таб Sin 0,0980
x = 0,20: исходный Sin = 0,1987, таб Sin 0,1960
x = 0,30: исходный Sin = 0,2955, таб Sin 0,2939
x = 0,40: исходный Sin = 0,3894, таб Sin 0,3859
x = 0,50: исходный Sin = 0,4794, таб Sin 0,4721
x = 0,60: исходный Sin = 0,5646, таб Sin 0,5582
x = 0,70: исходный Sin = 0,6442, таб Sin 0,6440
x = 0,80: исходный Sin = 0,7174, таб Sin 0,7079
x = 0,90: исходный Sin = 0,7833, таб Sin 0,7719
x = 1,00: исходный Sin = 0,8415, таб Sin 0,8358
x = 1,10: исходный Sin = 0,8912, таб Sin 0,8840
x = 1,20: исходный Sin = 0,9320, таб Sin 0,9180
x = 1,30: исходный Sin = 0,9636, таб Sin 0,9521
x = 1,40: исходный Sin = 0,9854, таб Sin 0,9848
x = 1,50: исходный Sin = 0,9975, таб Sin 0,9848
x = 1,60: исходный Sin = 0,9996, таб Sin 0,9848
x = 1,70: исходный Sin = 0,9917, таб Sin 0,9848

```

$\sin^2 + \cos^2 = 1$

```

== Тестирование суммы аналогов синуса и косинуса ==
x=0,00: sin^2 + cos^2 = 1,0000
x=0,10: sin^2 + cos^2 = 0,9753
x=0,20: sin^2 + cos^2 = 0,9705
x=0,30: sin^2 + cos^2 = 0,9854
x=0,40: sin^2 + cos^2 = 0,9850
x=0,50: sin^2 + cos^2 = 0,9704
x=0,60: sin^2 + cos^2 = 0,9756
x=0,70: sin^2 + cos^2 = 0,9994
x=0,80: sin^2 + cos^2 = 0,9751
x=0,90: sin^2 + cos^2 = 0,9706
x=1,00: sin^2 + cos^2 = 0,9859
x=1,10: sin^2 + cos^2 = 0,9845

```

## Анализ результата тождества при разном количестве точек

```
-- Изменение результирующая функции при изменении кол-ва точек ==
Точек: 5, погрешность: 0,14639599
Точек: 20, погрешность: 0,00681713
Точек: 50, погрешность: 0,00102635

-- Экспонента и работа с файлом --
```

## Работа с файлом, экспонентой и сравнение значений исходной функции и полученной из файла

```
-- Экспонента и работа с файлом ==
Сравнение значений исходной и считанной функций
Исходный у = 1,00, прочитанный у = 1,00
Исходный у = 2,72, прочитанный у = 2,72
Исходный у = 7,39, прочитанный у = 7,39
Исходный у = 20,09, прочитанный у = 20,09
Исходный у = 54,60, прочитанный у = 54,60
Исходный у = 148,41, прочитанный у = 148,41
Исходный у = 403,43, прочитанный у = 403,43
Исходный у = 1096,63, прочитанный у = 1096,63
Исходный у = 2980,96, прочитанный у = 2980,96
Исходный у = 8103,08, прочитанный у = 8103,08
```

## Работа с файлом, логарифмом и сравнение значений исходной функции и полученной из файла

```
== Логарифм и работа с файлом ==
сравнение
Исходный у = -2,30, прочитанный у = -2,30
Исходный у = 0,09, прочитанный у = 0,09
Исходный у = 0,73, прочитанный у = 0,73
Исходный у = 1,12, прочитанный у = 1,12
Исходный у = 1,40, прочитанный у = 1,40
Исходный у = 1,62, прочитанный у = 1,62
Исходный у = 1,80, прочитанный у = 1,80
Исходный у = 1,95, прочитанный у = 1,95
Исходный у = 2,08, прочитанный у = 2,08
Исходный у = 2,20, прочитанный у = 2,20
```

Сравнив бинарный и текстовый форматы, могу отметить, что преимущества первого в меньшем размере и высокой скорости работы, а второго в его читаемости и легкости редактирования

## Задание №9

При выполнении этого задания я изучила интерфейсы [java.io.Serializable](#) и [java.io.Externalizable](#). Могу отметить, что для реализации первого практически ничего не нужно: он сам сериализует всю программу, а также медленную работу, хранение большого количества служебной информации . Реализация второго ложится на плечи разработчика, но при этом это становится контролируемым процессом.

Я выбрала Externalizable, поэтому описала дополнительные методы

```
@Override  
public void writeExternal(ObjectOutput out) throws IOException {  
    out.writeInt(pointsCount);  
    // записываем все точки  
    FunctionNode current = head.getNext();  
    while (current != head) {  
        out.writeObject(current.getPoint());  
        current = current.getNext();  
    }  
}  
  
@Override  
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {  
  
    pointsCount = in.readInt();  
    this.head = new FunctionNode(point: null);  
    this.head.setPrev(head);  
    this.head.setNext(head);  
  
    for (int i = 0; i < pointsCount; i++) {  
        FunctionPoint point = (FunctionPoint) in.readObject();  
        addNodeToTail(point);  
    }  
}
```

```
@Override  
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());  
    }  
}  
  
@Override  
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {  
    this.pointsCount = in.readInt();  
    this.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);  
    }  
}
```

```
== Сериализация ==  
сравнение исходной и полученной функции  
Исходный у = NaN, полученный у = NaN  
Исходный у = 1,00, полученный у = 1,00  
Исходный у = 2,00, полученный у = 2,00  
Исходный у = 3,00, полученный у = 3,00  
Исходный у = 4,00, полученный у = 4,00  
Исходный у = 5,00, полученный у = 5,00  
Исходный у = 6,00, полученный у = 6,00  
Исходный у = 7,00, полученный у = 7,00  
Исходный у = 8,00, полученный у = 8,00  
Исходный у = 9,00, полученный у = 9,00  
Исходный у = 10,00, полученный у = 10,00
```