

# **Лабораторная работа №4**

Автор: Ежова Екатерина

Группа: 6203-010302D

## Задание 1

В классы `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` я добавила 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`. Чтобы точки были упорядочены по значению абсциссы, каждый раз сравниваем значение текущей точки со значением предыдущей. В массив добавляем копию точки, чтобы не нарушить инкапсуляцию.

```
public ArrayTabulatedFunction(FunctionPoint[] values) { 4 usages new *
    arrayLen = values.length;
    if (arrayLen < 2) {
        throw new IllegalArgumentException("Точек меньше двух");
    }
    double leftX = values[0].getX(); //получаем первое значение x
    arrayPoints = new FunctionPoint[arrayLen];
    arrayPoints[0] = new FunctionPoint(values[0]); //добавляем в массив
    double rightX;
    for (int i = 1; i < arrayLen; i++) { //с помощью цикла заполняем массив, каждый
        rightX = values[i].getX(); //присваиваем значение правой границе
        if (leftX >= rightX) {
            throw new IllegalArgumentException("Левая граница больше правой");
        }
        leftX = rightX;
        arrayPoints[i] = new FunctionPoint(values[i]); //добавляем копию
    }
}
```

```
public LinkedListTabulatedFunction(FunctionPoint[] values) { 2 usages new *
    len = values.length;
    if (len < 2) {
        throw new IllegalArgumentException("Точек меньше двух");
    }
    double leftX = values[0].getX(); //получаем первое значение x
    addNodeToTail(new FunctionPoint(values[0])); //добавляем в список
    double rightX;
    for (int i = 1; i < len; i++) { //с помощью цикла заполняем список, каждый раз пров
        rightX = values[i].getX(); //присваиваем значение правой границе
        if (leftX >= rightX) {
            throw new IllegalArgumentException("Левая граница больше правой");
        }
        leftX = rightX;
        addNodeToTail(new FunctionPoint(values[i])); //добавляем копию
    }
}
```

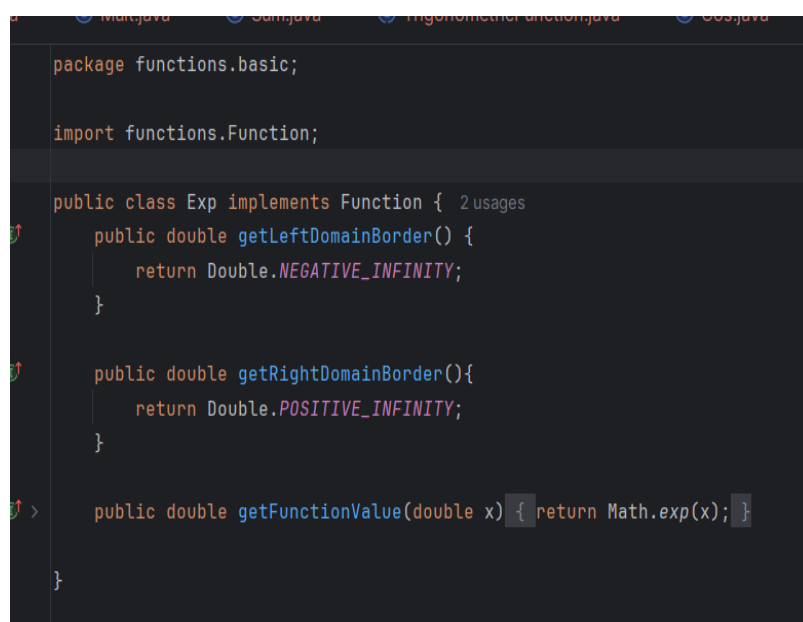
## Задание 2

В пакете `functions` я создала интерфейс `Function`, описывающий функции одной переменной и содержащий следующие методы: `public double getLeftDomainBorder()`, `public double getRightDomainBorder()`, `public double getFunctionValue(double x)`. Затем я исключила соответствующие методы из интерфейса `TabulatedFunction` и с помощью ключевого слова `extends` расширила интерфейс `Function`.

## Задание 3

В этом задании я создала пакет `functions.basic`. В этом пакете находится 5 публичных классов функций, заданных аналитически и абстрактный класс `TrigonometricFunction`, реализующий интерфейс `Function` и описывающий методы получения границ области определения для тригонометрических функций. Классы `Sin`, `Cos` и `Tan` наследуют от этого класса и реализуют метод `getFunctionValue(double x)` каждый для своего класса. В пакете есть Классы `Exp`, объекты которого вычисляют значение экспоненты, и `Log`, объекты которого вычисляют значение логарифма по заданному основанию. Так как метод `Math.log()` вычисляет логарифм по натуральному основанию, в методе `getFunctionValue()` воспользуемся формулой для смены основания логарифма. Также все эти классы реализуют интерфейс `Function`.

Пример класса для 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); }
}
```

В качестве области определения берем +- бесконечность.

## Задание 4

В этом задании я создала пакет `functions.meta`. В этом пакете 6 публичных классов, реализующих интерфейс `Function` и позволяющий комбинировать функции. В классе `Sum` есть 2 приватных поля, хранящих объекты каждой функции, в конструкторе получаем ссылки типа `Function` на объекты суммируемых функций.левой границей области определения является наибольшая левая граница одной из функций, а правой границей – наименьшая. В методе `getFunctionValue()` делаем проверку на то, принадлежит ли абсцисса области определения, если нет - возвращаем значение `Double.NaN`. Класс `Mult` аналогично, только вместо суммы в методе `getFunctionValue()` возвращаем произведение значений функций в этой точке. Класс `Power` имеет два приватных поля, хранящих функцию и степень, в которую нужно ее возвести. С помощью метода `Math.pow()` возвращаем значение функции в точке, возведенное в степень. В конструктор класса `Scale` передается функция и коэффициенты для масштабирования по оси `x` и `y`. В области определения умножаем левую и правую границу на коэффициент масштабирования по оси абсцисс, если коэффициент отрицательный левая и правая граница меняются местами. Возвращаем новое значение функции. Аналогично класс `Shift`, только его конструктор в качестве параметров принимает значения для сдвига по оси координат, область определения сдвигается на значение сдвига по оси абсцисс, а затем возвращается новое значение функции. Класс `Composition` хранит два приватных поля с 2 функциями, Область определения берем у второй функции, то есть у той, которая внутренняя, Возвращаем значение первой функции от значения, которое принимает вторая функция в точке `x`.

Класс `Composition`:

```
package functions.meta;

import functions.Function;

public class Composition implements Function { 2 usages
    private Function fun1; 2 usages
    private Function fun2; 4 usages

    public Composition(Function fun1, Function fun2) { 4 usages
        this.fun1 = fun1;
        this.fun2 = fun2;
    }

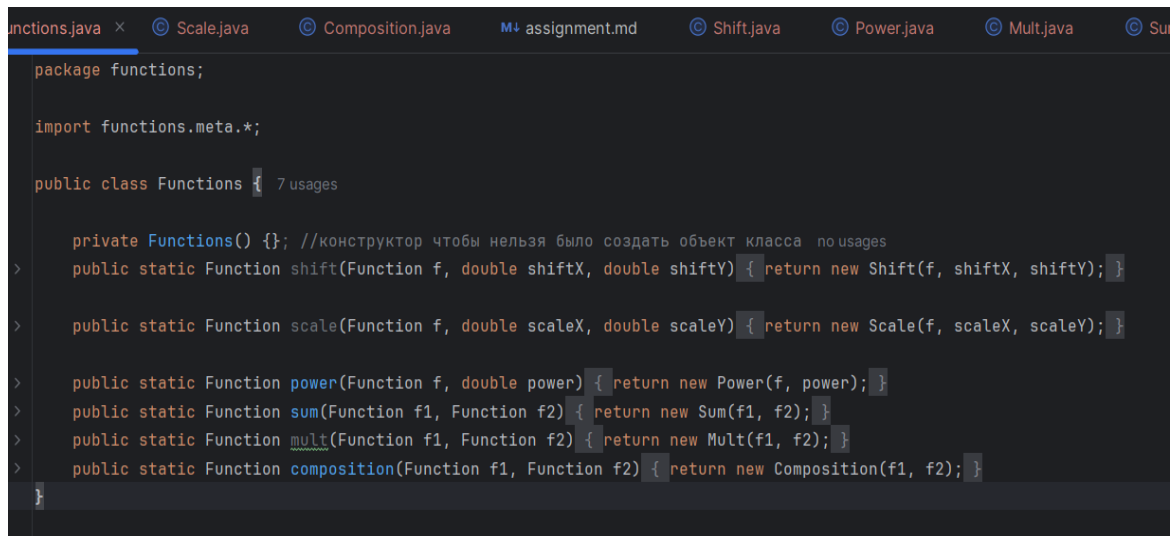
    public double getLeftDomainBorder() {
        return fun2.getLeftDomainBorder();
    }

    public double getRightDomainBorder() { return fun2.getRightDomainBorder(); }

    public double getFunctionValue(double x) {
        if (x >= getLeftDomainBorder() && x <= getRightDomainBorder()) { //входит ли и
            return fun1.getFunctionValue(fun2.getFunctionValue(x));
        }
        return Double.NaN; //результата не существует
    }
}
```

## Задание 5

В пакете functions я создала класс Functions, в котором есть приватный конструктор, чтобы нельзя было создать объекты этого класса, и статические методы:



```
package functions;

import functions.meta.*;

public class Functions { 7 usages

    private Functions() {}; //конструктор чтобы нельзя было создать объект класса no usages
    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) { 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); }
```

Каждый метод возвращает объект типа классов из прошлого задания.

## Задание 6

В пакете functions я создала класс TabulatedFunctions, в котором есть приватный конструктор, чтобы нельзя было создать объекты этого класса и 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:

```
public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) { 7 usages
    if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
        throw new IllegalArgumentException("Границы для табулирования выходят за область определения функции");
    }
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Точек меньше двух");
    }
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница больше правой");
    }
    double[] values = new double[pointsCount]; //создали массив, который будет хранить значения табулированной функции
    double funVal; //переменная для хранения y в каждой точке
    double intervalLength = Math.abs(leftX - rightX) / (pointsCount - 1); //находим длину интервала между двумя точками
    for (int i=0; i < pointsCount; i++) {
        if (i != (pointsCount - 1)) { //если не последняя точка
            funVal = function.getFunctionValue(leftX + i*intervalLength); //получаем значение функции в заданной точке, каждый раз перемещая x на длину интервала
        }
        else {
            funVal = function.getFunctionValue(rightX); //последней точке присваиваем значение y правой границы
        }
        values[i] = funVal; //заполняем массив значениями
    }
    TabulatedFunction tabFun = new ArrayTabulatedFunction(leftX, rightX, values); //создаем объект, который будет хранить табулированную функцию в виде массива, в конст
    return tabFun;
}
```

В методе я создаю массив для хранения значений функции в каждой точке разбиения и затем создаю объект, где в конструктор передаю эти значения, и возвращаю новую табулированную функцию.

## Задание 7

Также в классе TabulatedFunctions реализовала следующие методы:

```
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException { 1 usage
    try (DataOutputStream Out = new DataOutputStream(out)) {
        Out.writeInt(function.getPointsCount());
        for (int i=0; i < function.getPointsCount(); i++) {
            Out.writeDouble(function.getPointX(i));
            Out.writeDouble(function.getPointY(i));
        }
        Out.flush(); //сбрасываем буфер
    }
}

public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException, InappropriateFunctionPointException { 1 usage
    try (DataInputStream In = new DataInputStream(in)) {
        int pointsCount = In.readInt();
        ArrayTabulatedFunction tabFun = new ArrayTabulatedFunction( leftX: 0, rightX: pointsCount - 1, pointsCount); //создаем объект для хранения функции
        for (int i=0; i < pointsCount; i++) {
            tabFun.setPointX(i, In.readDouble()); //записываем значения x и y
            tabFun.setPointY(i, In.readDouble());
        }

        return tabFun;
    }
}
```

```
@
public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException { 1 usage
    try (BufferedWriter Out = new BufferedWriter(out)) {
        Out.write(String.valueOf(function.getPointsCount()));
        Out.newLine();
        for (int i=0; i < function.getPointsCount(); i++) {
            Out.write(String.valueOf(function.getPointX(i)));
            Out.write(" ");
            Out.write(String.valueOf(function.getPointY(i)));
            Out.newLine();
        }
        Out.flush();
    }
}

@
public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException, InappropriateFunctionPointException { 1 us
    StreamTokenizer In = new StreamTokenizer(in);
    In.nextToken(); //переход к следующему токenu
    int pointsCount = (int)In.nval; //числовое значение токена преобразуем в число типа int
    ArrayTabulatedFunction tabFun = new ArrayTabulatedFunction( leftX: 0, rightX: pointsCount - 1, pointsCount);
    for (int i=0; i < pointsCount; i++) {
        In.nextToken();
        tabFun.setPointX(i, In.nval); //записываем значения x и y
        In.nextToken();
        tabFun.setPointY(i, In.nval); //In.nval - статическое поле типа double
    }

    return tabFun;
}
```

Методы вывода и ввода в байтовый поток и методы вывода и ввода в символьный поток.

Возникающее исключение перебрасываем на более верхний уровень и обрабатываем там, внутри методов закрываются только потоки обертки, исходные потоки необходимо закрывать только там, где их открыли.



## Задание 8

```
C:\Program Files\Java\jdk-20\bin>java.exe -jar javadgen-1.0-11\Program Files\Jett
Sin на отрезке (x = 0.0) = 0.0
Cos на отрезке (x = 0.0) = 1.0
Sin на отрезке (x = 0.1) = 0.09983341664682815
Cos на отрезке (x = 0.1) = 0.9950041652780258
Sin на отрезке (x = 0.2) = 0.19866933079506122
Cos на отрезке (x = 0.2) = 0.9800665778412416
Sin на отрезке (x = 0.30000000000000004) = 0.2955202066613396
Cos на отрезке (x = 0.30000000000000004) = 0.955336489125606
Sin на отрезке (x = 0.4) = 0.3894183423086505
Cos на отрезке (x = 0.4) = 0.9210609940028851
Sin на отрезке (x = 0.5) = 0.479425538604203
Cos на отрезке (x = 0.5) = 0.8775825618903728
Sin на отрезке (x = 0.6) = 0.5646424733950354
Cos на отрезке (x = 0.6) = 0.8253356149096783
Sin на отрезке (x = 0.7) = 0.644217687237691
Cos на отрезке (x = 0.7) = 0.7648421872844885
Sin на отрезке (x = 0.7999999999999999) = 0.7173560908995227
Cos на отрезке (x = 0.7999999999999999) = 0.6967067093471655
Sin на отрезке (x = 0.8999999999999999) = 0.7833269096274833
Cos на отрезке (x = 0.8999999999999999) = 0.6216099682706645
Sin на отрезке (x = 0.9999999999999999) = 0.8414709848078964
Cos на отрезке (x = 0.9999999999999999) = 0.5403023058681398
Sin на отрезке (x = 1.0999999999999999) = 0.8912073600614353
Cos на отрезке (x = 1.0999999999999999) = 0.4535961214255775
Sin на отрезке (x = 1.2) = 0.9320390859672263
Cos на отрезке (x = 1.2) = 0.3623577544766736
Sin на отрезке (x = 1.3) = 0.963558185417193
Cos на отрезке (x = 1.3) = 0.26749882862458735
```

### Табулированные функции

```
Sin на отрезке (x = 0.0) = 0.0
Cos на отрезке (x = 0.0) = 1.0
Sin на отрезке (x = 0.1) = 0.09798155360510165
Cos на отрезке (x = 0.1) = 0.9827232084876878
Sin на отрезке (x = 0.2) = 0.1959631072102033
Cos на отрезке (x = 0.2) = 0.9654464169753757
Sin на отрезке (x = 0.30000000000000004) = 0.29394466081530496
Cos на отрезке (x = 0.30000000000000004) = 0.9481696254630635
Sin на отрезке (x = 0.4) = 0.38590680571121505
Cos на отрезке (x = 0.4) = 0.91435464443779
Sin на отрезке (x = 0.5) = 0.47207033789781927
Cos на отрезке (x = 0.5) = 0.864608105941514
Sin на отрезке (x = 0.6) = 0.5582338700844235
Cos на отрезке (x = 0.6) = 0.8148615674392491
Sin на отрезке (x = 0.7) = 0.6439824415274444
Cos на отрезке (x = 0.7) = 0.7646204979800333
Sin на отрезке (x = 0.7999999999999999) = 0.707935358675545
Cos на отрезке (x = 0.7999999999999999) = 0.6884043792119047
Sin на отрезке (x = 0.8999999999999999) = 0.7718882758236456
Cos на отрезке (x = 0.8999999999999999) = 0.6121882604437761
Sin на отрезке (x = 0.9999999999999999) = 0.8358411929717461
Cos на отрезке (x = 0.9999999999999999) = 0.5359721416756473
Sin на отрезке (x = 1.0999999999999999) = 0.8839933571281424
Cos на отрезке (x = 1.0999999999999999) = 0.45063345391435955
Sin на отрезке (x = 1.2) = 0.9180219935851436
Cos на отрезке (x = 1.2) = 0.35714054363391856
```

### Сумма квадратов синуса и косинуса(10 точек разбиения)

```
Сумма на отрезке (x = 0.0) = 1.0
Сумма на отрезке (x = 0.1) = 0.9753452893472049
Сумма на отрезке (x = 0.2) = 0.9704883234380686
Сумма на отрезке (x = 0.30000000000000004) = 0.9854291022725908
Сумма на отрезке (x = 0.4) = 0.9849684785101429
Сумма на отрезке (x = 0.5) = 0.9703975807827336
Сумма на отрезке (x = 0.6) = 0.975624427798983
Сумма на отрезке (x = 0.7) = 0.9993578909268825
Сумма на отрезке (x = 0.7999999999999999) = 0.9750730613812004
Сумма на отрезке (x = 0.8999999999999999) = 0.9705859765791769
Сумма на отрезке (x = 0.9999999999999999) = 0.9858966365208117
Сумма на отрезке (x = 1.0999999999999999) = 0.9845147652334687
Сумма на отрезке (x = 1.2) = 0.9703137486131723
Сумма на отрезке (x = 1.3) = 0.9759104767365345
Сумма на отрезке (x = 1.4000000000000001) = 0.9987226923395387
Сумма на отрезке (x = 1.5000000000000002) = 0.9748077439009694
Сумма на отрезке (x = 1.6000000000000003) = 0.9706905402060587
Сумма на отрезке (x = 1.7000000000000004) = 0.9863710812548067
Сумма на отрезке (x = 1.8000000000000005) = 0.9840679624425679
Сумма на отрезке (x = 1.9000000000000006) = 0.9702368269293845
Сумма на отрезке (x = 2.0000000000000004) = 0.9762034361598597
Сумма на отрезке (x = 2.1000000000000005) = 0.9980944042379682
Сумма на отрезке (x = 2.2000000000000006) = 0.9745493369065118
Сумма на отрезке (x = 2.3000000000000007) = 0.9708020143187142
Сумма на отрезке (x = 2.4000000000000001) = 0.9868524364745752
Сумма на отрезке (x = 2.5000000000000001) = 0.9836280701374409
Сумма на отрезке (x = 2.6000000000000001) = 0.9701668157313703
```

2025. > Main > main

```

Сумма квадратов синуса и косинуса(100 точек разбиения)

Сумма на отрезке (x = 0.0) = 1.0
Сумма на отрезке (x = 0.1) = 0.9998707262777297
Сумма на отрезке (x = 0.2) = 0.9997875329627255
Сумма на отрезке (x = 0.30000000000000004) = 0.9997504200549877
Сумма на отрезке (x = 0.4) = 0.9997593875545162
Сумма на отрезке (x = 0.5) = 0.999814435461311
Сумма на отрезке (x = 0.6) = 0.9999155637753722
Сумма на отрезке (x = 0.7) = 0.9999442080013095
Сумма на отрезке (x = 0.7999999999999999) = 0.9998328692780962
Сумма на отрезке (x = 0.8999999999999999) = 0.9997676109621492
Сумма на отрезке (x = 0.9999999999999999) = 0.9997484330534682
Сумма на отрезке (x = 1.0999999999999999) = 0.9997753355520536
Сумма на отрезке (x = 1.2) = 0.9998483184579052
Сумма на отрезке (x = 1.3) = 0.9999673817710231
Сумма на отрезке (x = 1.4000000000000001) = 0.9998953965006281
Сумма на отрезке (x = 1.5000000000000002) = 0.9998019927764716
Сумма на отрезке (x = 1.6000000000000003) = 0.9997546694595814
Сумма на отрезке (x = 1.7000000000000004) = 0.9997534265499576
Сумма на отрезке (x = 1.8000000000000005) = 0.9997982640476
Сумма на отрезке (x = 1.9000000000000006) = 0.9998891819525084
Сумма на отрезке (x = 2.0000000000000004) = 0.9999751146303215
Сумма на отрезке (x = 2.1000000000000005) = 0.9998535654979555
Сумма на отрезке (x = 2.2000000000000006) = 0.999778096772856
Сумма на отрезке (x = 2.3000000000000007) = 0.9997487084550228
Сумма на отрезке (x = 2.4000000000000001) = 0.9997654005444558
Сумма на отрезке (x = 2.5000000000000001) = 0.9998281730411551
Сумма на отрезке (x = 2.6000000000000001) = 0.9999370259451208

```

Чем больше точек разбиения, тем результат ближе к единице.

#### Экспонента

```

Отрезок (x = 0) ex1 = 1.0 ex2 = 1.0
Отрезок (x = 1) ex1 = 2.718281828459045 ex2 = 2.718281828459045
Отрезок (x = 2) ex1 = 7.38905609893065 ex2 = 7.38905609893065
Отрезок (x = 3) ex1 = 20.085536923187668 ex2 = 20.085536923187668
Отрезок (x = 4) ex1 = 54.598150033144236 ex2 = 54.59815003314424
Отрезок (x = 5) ex1 = 148.4131591025766 ex2 = 148.4131591025766
Отрезок (x = 6) ex1 = 403.4287934927351 ex2 = 403.4287934927351
Отрезок (x = 7) ex1 = 1096.6331584284585 ex2 = 1096.6331584284585
Отрезок (x = 8) ex1 = 2980.9579870417283 ex2 = 2980.9579870417283
Отрезок (x = 9) ex1 = 8103.083927575384 ex2 = 8103.083927575384
Отрезок (x = 10) ex1 = 22026.465794806718 ex2 = 22026.46579480672

```

#### Логарифм

```

Отрезок (x = 0) log1 = NaN log2 = NaN
Отрезок (x = 1) log1 = 0.0 log2 = 0.0
Отрезок (x = 2) log1 = 0.6931471805599453 log2 = 0.6931471805599453
Отрезок (x = 3) log1 = 1.0986122886681098 log2 = 1.0986122886681098
Отрезок (x = 4) log1 = 1.3862943611198906 log2 = 1.3862943611198906
Отрезок (x = 5) log1 = 1.6094379124341003 log2 = 1.6094379124341003
Отрезок (x = 6) log1 = 1.791759469228055 log2 = 1.791759469228055
Отрезок (x = 7) log1 = 1.9459101490553132 log2 = 1.9459101490553132
Отрезок (x = 8) log1 = 2.0794415416798357 log2 = 2.0794415416798357
Отрезок (x = 9) log1 = 2.1972245773362196 log2 = 2.1972245773362196
Отрезок (x = 10) log1 = 2.302585092994046 log2 = 2.302585092994046

```

## Задание 9

### Сериализация

```
Отрезок (x = 0) Fun = 0.0 Fun1 = 0.0
Отрезок (x = 1) Fun = 1.0 Fun1 = 1.0
Отрезок (x = 2) Fun = 2.0 Fun1 = 2.0
Отрезок (x = 3) Fun = 3.0 Fun1 = 3.0
Отрезок (x = 4) Fun = 4.0 Fun1 = 4.0
Отрезок (x = 5) Fun = 5.0 Fun1 = 5.0
Отрезок (x = 6) Fun = 6.0 Fun1 = 6.0
Отрезок (x = 7) Fun = 7.0 Fun1 = 7.0
Отрезок (x = 8) Fun = 8.0 Fun1 = 8.0
Отрезок (x = 9) Fun = 9.0 Fun1 = 9.0
Отрезок (x = 10) Fun = 10.0 Fun1 = 10.0
```

Преимущества Serializable: простота и автоматизм Недостатки: скорость и размер файла

Преимущества Externalizable: размер, скорость, безопасность. Недостатки: сложность написания

Реализация Externalizable:

```
package functions;
import java.io.*;
public class ArrayTabulatedFunction implements TabulatedFunction, Externalizable{
    private FunctionPoint[] arrayPoints; //массив, хранящий точки функции
    private int arrayLen; //длина массива

    public ArrayTabulatedFunction() {} //конструктор без параметров

    @Override
    public void writeExternal(ObjectOutput out) throws IOException {
        out.writeInt(arrayLen);
        for (int i = 0; i < arrayLen; i++) {
            out.writeDouble(arrayPoints[i].getx());
            out.writeDouble(arrayPoints[i].gety());
        }
    }

    @Override
    public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
        arrayLen = in.readInt();
        arrayPoints = new FunctionPoint[arrayLen];
        for (int i = 0; i < arrayLen; i++) {
            arrayPoints[i] = new FunctionPoint(in.readDouble(), in.readDouble());
        }
    }
}
```

Создали конструктор без параметров и перезаписали методы для вывода и ввода.

writeObject(Fun) сериализует объект в файл, in.readObject() десериализует объект из файла