Лабораторная работа №4

Выполнил : Горынин Дмитрий, ст.гр. 6201-120303D.

Задание 1.

В классах ArrayTabulatedFunction и LinkedListTabulatedFunction добавьте конструкторы, получающие сразу все точки функции в виде массива объектов типа FunctionPoint.

```
public ArrayTabulatedFunction(FunctionPoint[] pointsArray) {  1 usage
    if (pointsArray == null) {
        throw new IllegalArgumentException("Points array cannot be null");
    }
    if (pointsArray.length < 2) {
        throw new IllegalArgumentException("Points array must contain at least 2 points");
    }

    // Проверяем упорядоченность точек по x
    for (int i = 1; i < pointsArray.length; i++) {
        if (pointsArray[i] == null || pointsArray[i-1] == null) {
            throw new IllegalArgumentException("Points array cannot contain null elements");
        }
        if (pointsArray[i].getX() <= pointsArray[i-1].getX()) {
            throw new IllegalArgumentException(
                    "Points must be strictly increasing by x. " +
                            "Point " + i + " has x=" + pointsArray[i].getX() +
                            " which is not greater than point " + (i-1) +
                            " with x=" + pointsArray[i-1].getX()
            );
        }
    }
```

```java
public LinkedListTabulatedFunction(FunctionPoint[] pointsArray) {  no usages
    if (pointsArray == null) {
        throw new IllegalArgumentException("Points array cannot be null");
    }
    if (pointsArray.length < 2) {
        throw new IllegalArgumentException("Points array must contain at least 2 points");
    }

    for (int i = 1; i < pointsArray.length; i++) {
        if (pointsArray[i] == null || pointsArray[i-1] == null) {
            throw new IllegalArgumentException("Points array cannot contain null elements");
        }
        if (pointsArray[i].getX() <= pointsArray[i-1].getX()) {
            throw new IllegalArgumentException(
                    "Points must be strictly increasing by x. " +
                            "Point " + i + " has x=" + pointsArray[i].getX() +
                            " which is not greater than point " + (i-1) +
                            " with x=" + pointsArray[i-1].getX()
            );
        }
    }

    this.size = pointsArray.length;

    head = new Node(pointsArray[0]);
    Node current = head;

    for (int i = 1; i < pointsArray.length; i++) {
        Node newNode = new Node(pointsArray[i]);
        current.next = newNode;
        newNode.prev = current;
        current = newNode;
    }
    tail = current;
}
```

Invalid

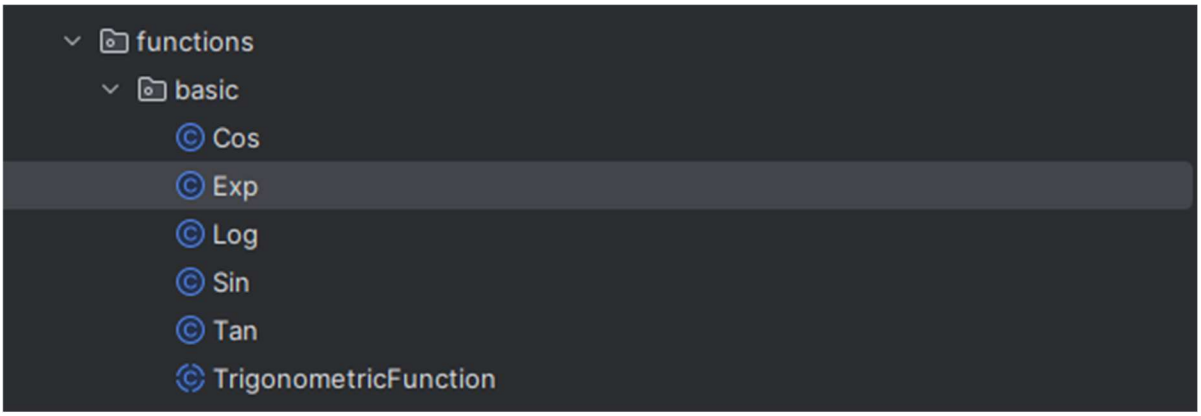
Задание 2.

В пакете functions создайте интерфейс Function

```java
package functions;

/**
 * Интерфейс для функций одной переменной
 */
public interface Function {

    /**
     * Возвращает значение левой границы области определения функции
     * @return левая граница области определения
     */
    double getLeftDomainBorder();  15 implementations

    /**
     * Возвращает значение правой границы области определения функции
     * @return правая граница области определения
     */
    double getRightDomainBorder();  15 implementations

    /**
     * Возвращает значение функции в заданной точке
     * @param x точка, в которой вычисляется значение функции
     * @return значение функции в точке x
     */
    double getFunctionValue(double x);  18 implementations
}
```

Задание 3.

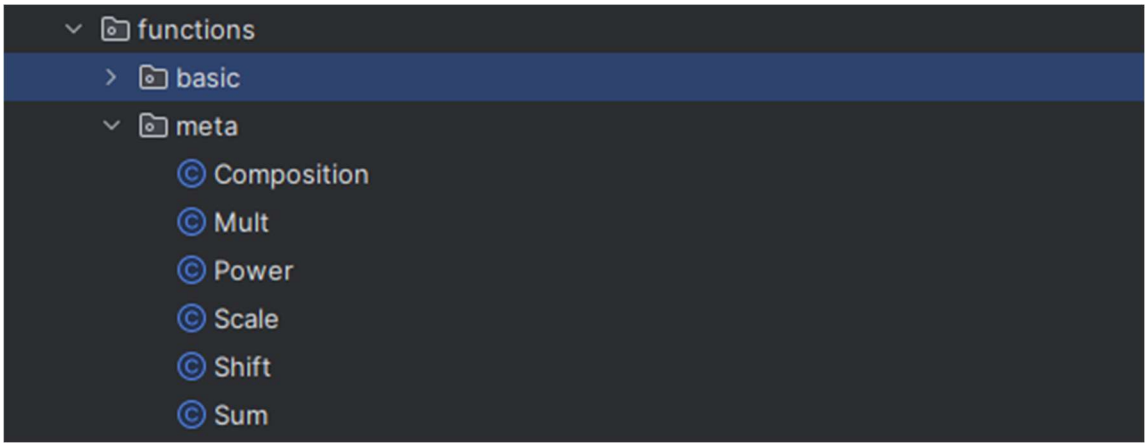
Создайте пакет functions.basic, в нём будут описаны классы ряда функций, заданных аналитически.



```java
package functions.basic;

import functions.Function;

/**
 * Класс для вычисления экспоненциальной функции f(x) = e^x
 */
public class Exp implements Function {  6 usages

    /**
     * Конструктор по умолчанию
     */
    public Exp() {  5 usages
        // Нет необходимости в инициализации
    }

    /**
     * Возвращает левую границу области определения
     * @return Double.NEGATIVE_INFINITY (минус бесконечность)
     */
    @Override
    public double getLeftDomainBorder() {
        return Double.NEGATIVE_INFINITY;
    }

    /**
     * Возвращает правую границу области определения
     * @return Double.POSITIVE_INFINITY (плюс бесконечность)
     */
    @Override
    public double getRightDomainBorder() {
        return Double.POSITIVE_INFINITY;
    }

    /**
     * Вычисляет значение экспоненты в заданной точке
     * @param x точка, в которой вычисляется значение
     * @return e^x
     */
    @Override
    public double getFunctionValue(double x) {
        // Экспонента определена для всех действительных чисел
```

Задание 4.

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




```java
package functions.meta;

import functions.Function;

/**
 * Класс для представления суммы двух функций: f(x) = g(x) + h(x)
 */
public class Sum implements Function {  1 usage
    private final Function first;  6 usages
    private final Function second;  6 usages

    /**
     * Конструктор суммы двух функций
     * @param first первая функция
     * @param second вторая функция
     */
    public Sum(Function first, Function second) {  1 usage
        if (first == null || second == null) {
            throw new IllegalArgumentException("Функции не могут быть null");
        }
        this.first = first;
        this.second = second;
    }

    /**
     * Возвращает левую границу области определения
     * @return максимум левых границ двух функций
     */
    @Override
    public double getLeftDomainBorder() {
        return Math.max(first.getLeftDomainBorder(), second.getLeftDomainBorder());
    }

    /**
     * Возвращает правую границу области определения
     * @return минимум правых границ двух функций
     */
    @Override
    public double getRightDomainBorder() {
        return Math.min(first.getRightDomainBorder(), second.getRightDomainBorder());
```

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.

```java
package functions;

import functions.meta.*;

/**
 * Утилитный класс для работы с функциями.
 * Содержит статические методы для создания мета-функций.
 * Не может быть инстанциирован.
 */
public class Functions {

    /**
     * Приватный конструктор для предотвращения создания объектов класса.
     */
    private Functions() {  no usages
        throw new AssertionError( detailMessage: "Нельзя создавать объекты утилитного класса Functions");
    }

    /**
     * Возвращает функцию, полученную из исходной сдвигом вдоль осей.
     * @param f исходная функция
     * @param shiftX сдвиг вдоль оси X
     * @param shiftY сдвиг вдоль оси Y
     * @return сдвинутая функция: f(x) = shiftY + f(x + shiftX)
     * @throws IllegalArgumentException если исходная функция равна null
     */
    public static Function shift(Function f, double shiftX, double shiftY) {  1 usage
        if (f == null) {
            throw new IllegalArgumentException("Исходная функция не может быть null");
        }
        return new Shift(f, shiftX, shiftY);
    }

    /**
     * Возвращает функцию, полученную из исходной масштабированием вдоль осей.
     * @param f исходная функция
     * @param scaleX коэффициент масштабирования вдоль оси X
     * @param scaleY коэффициент масштабирования вдоль оси Y
     * @return масштабированная функция: f(x) = scaleY * f(scaleX * x)
     * @throws IllegalArgumentException если исходная функция равна null
```

Задание 6.

В пакете functions создайте класс TabulatedFunctions, содержащий вспомогательные статические методы для работы с табулированными функциями.

```java
package functions;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

/**
 * Утилитный класс для работы с табулированными функциями.
 * Содержит статические методы для создания, преобразования и сериализации табулированных функций.
 * Не может быть инстанциирован.
 */
public class TabulatedFunctions {

    /**
     * Приватный конструктор для предотвращения создания объектов класса.
     */
    private TabulatedFunctions() {  no usages
        throw new AssertionError( detailMessage: "Нельзя создавать объекты утилитного класса TabulatedFunctions");
    }

    // ==================== Методы для создания табулированных функций ====================

    /**
     * Табулирует функцию на заданном отрезке с заданным количеством точек.
     *
     * @param function функция для табулирования
     * @param leftX левая граница отрезка табулирования
     * @param rightX правая граница отрезка табулирования
     * @param pointsCount количество точек табулирования (должно быть >= 2)
     * @return табулированная функция
     * @throws IllegalArgumentException если function равна null
     * @throws IllegalArgumentException если pointsCount < 2
     * @throws IllegalArgumentException если leftX >= rightX
     * @throws IllegalArgumentException если границы табулирования выходят за область определения функции
     */
    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {  7 usages
        // Проверка входных параметров
        if (function == null) {
            throw new IllegalArgumentException("Функция не может быть null");
        }
        if (pointsCount < 2) {
```

Задание 7.

В класс TabulatedFunctions добавьте следующие методы.

```java
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException {  1 usage
    if (function == null) {
        throw new NullPointerException("Функция не может быть null");
    }
    if (out == null) {
        throw new NullPointerException("Выходной поток не может быть null");
    }

    DataOutputStream dataOut = new DataOutputStream(out);

    // Записываем количество точек
    int pointsCount = function.getPointsCount();
    dataOut.writeInt(pointsCount);

    // Записываем координаты всех точек
    for (int i = 0; i < pointsCount; i++) {
        dataOut.writeDouble(function.getPointX(i));
        dataOut.writeDouble(function.getPointY(i));
    }

    // Принудительно сбрасываем буфер
    dataOut.flush();
    // Не закрываем dataOut, чтобы не закрывать переданный поток out
}
```

Проверка написанного кода и вывод main:

```
1. Тестирование Sin и Cos:
=========================
sin(x) и cos(x) на отрезке [0, π] с шагом 0.1:
x          sin(x)          cos(x)
-----------------------------------------------
0.0000     0.0000000000    1.0000000000
0.1000     0.0998334166    0.9950041653
0.2000     0.1986693308    0.9800665778
0.3000     0.2955202067    0.9553364891
0.4000     0.3894183423    0.9210609940
0.5000     0.4794255386    0.8775825619
0.6000     0.5646424734    0.8253356149
0.7000     0.6442176872    0.7648421873
0.8000     0.7173560909    0.6967067093
0.9000     0.7833269096    0.6216099683
1.0000     0.8414709848    0.5403023059
1.1000     0.8912073601    0.4535961214
1.2000     0.9320390860    0.3623577545
1.3000     0.9635581854    0.2674988286
1.4000     0.9854497300    0.1699671429
1.5000     0.9974949866    0.0707372017
1.6000     0.9995736030    -0.0291995223
1.7000     0.9916648105    -0.1288444943
1.8000     0.9738476309    -0.2272020947
1.9000     0.9463000877    -0.3232895669
2.0000     0.9092974268    -0.4161468365
2.1000     0.8632093666    -0.5048461046
```

```
2.1000     0.8632093666    -0.5048461046
2.2000     0.8084964038    -0.5885011173
2.3000     0.7457052122    -0.6662760213
2.4000     0.6754631806    -0.7373937155
2.5000     0.5984721441    -0.8011436155
2.6000     0.5155013718    -0.8568887534
2.7000     0.4273798802    -0.9040721420
2.8000     0.3349881502    -0.9422223407
2.9000     0.2392493292    -0.9709581651
3.0000     0.1411200081    -0.9899924966
3.1000     0.0415806624    -0.9991351503

2. Табулированные аналоги Sin и Cos:
==================================
Сравнение точных и табулированных значений:
x          sin(x) точн.    sin(x) таб.     cos(x) точн.    cos(x) таб.
------------------------------------------------------------------------
0.0000     0.0000000000    0.0000000000    1.0000000000    1.0000000000
0.1000     0.0998334166    0.0979815536    0.9950041653    0.9827232085
0.2000     0.1986693308    0.1959631072    0.9800665778    0.9654464170
```

0.3000	0.2955202067	0.2939446608	0.9553364891	0.9481696255
0.4000	0.3894183423	0.3859068057	0.9210609940	0.9143546444
0.5000	0.4794255386	0.4720703379	0.8775825619	0.8646081059
0.6000	0.5646424734	0.5582338701	0.8253356149	0.8148615674
0.7000	0.6442176872	0.6439824415	0.7648421873	0.7646204980
0.8000	0.7173560909	0.7079353587	0.6967067093	0.6884043792
0.9000	0.7833269096	0.7718882758	0.6216099683	0.6121882604
1.0000	0.8414709848	0.8358411930	0.5403023059	0.5359721417
1.1000	0.8912073601	0.8839933571	0.4535961214	0.4506334539
1.2000	0.9320390860	0.9180219936	0.3623577545	0.3571405436
1.3000	0.9635581854	0.9520506300	0.2674988286	0.2636476334
1.4000	0.9854497300	0.9848077530	0.1699671429	0.1699305209
1.5000	0.9974949866	0.9848077530	0.0707372017	0.0704374439
1.6000	0.9995736030	0.9848077530	-0.0291995223	-0.0290556331
1.7000	0.9916648105	0.9848077530	-0.1288444943	-0.1285487101
1.8000	0.9738476309	0.9662040429	-0.2272020947	-0.2247614510
1.9000	0.9463000877	0.9321754065	-0.3232895669	-0.3182543613
2.0000	0.9092974268	0.8981467700	-0.4161468365	-0.4117472716
2.1000	0.8632093666	0.8624409083	-0.5048461046	-0.5042718354
2.2000	0.8084964038	0.7984879911	-0.5885011173	-0.5804879542
2.3000	0.7457052122	0.7345350740	-0.6662760213	-0.6567040730
2.4000	0.6754631806	0.6705821568	-0.7373937155	-0.7329201917
2.5000	0.5984721441	0.5940715695	-0.8011436155	-0.7941706620
2.6000	0.5155013718	0.5079080374	-0.8568887534	-0.8439172005
2.7000	0.4273798802	0.4217445052	-0.9040721420	-0.8936637390
2.8000	0.3349881502	0.3346977890	-0.9422223407	-0.9409837494
2.9000	0.2392493292	0.2367162354	-0.9709581651	-0.9582605409
3.0000	0.1411200081	0.1387346818	-0.9899924966	-0.9755373324
3.1000	0.0415806624	0.0407531282	-0.9991351503	-0.9928141240

```
Максимальные погрешности:
sin: 0,0147658500
cos: 0,0146201609

3. Сумма квадратов табулированных функций:
========================================
sin²(x) + cos²(x) должна быть равна 1 (теоретически)
Исследование влияния количества точек табуляции:
================================================

Количество точек табуляции: 5
x          sin²+cos²       погрешность
----------------------------------------
0.0000     1.0000000000    0.0000000000
Максимальная погрешность: 0,1463959903

Количество точек табуляции: 10
x          sin²+cos²       погрешность
----------------------------------------
0.0000     1.0000000000    0.0000000000
Максимальная погрешность: 0,0298331843

Количество точек табуляции: 20
x          sin²+cos²       погрешность
----------------------------------------
0.0000     1.0000000000    0.0000000000
Максимальная погрешность: 0,0068171324
```

```
Количество точек табуляции: 50
x          sin²+cos²       погрешность
----------------------------------------
0.0000     1.0000000000    0.0000000000
Максимальная погрешность: 0,0010263525

Количество точек табуляции: 100
x          sin²+cos²       погрешность
----------------------------------------
0.0000     1.0000000000    0.0000000000
Максимальная погрешность: 0,0002515669

4. Тестирование экспоненты с записью в файл:
========================================
Функция записана в файл: exp_function.txt
```

```
Сравнение исходной и считанной экспоненты:
x          исходная exp(x)      считанная            разница
-----------------------------------------------------------------
0          1.0000000000         1.0000000000         0.0000000000
1          2.7182818285         2.7182818285         0.0000000000
2          7.3890560989         7.3890560989         0.0000000000
3          20.0855369232        20.0855369232        0.0000000000
4          54.5981500331        54.5981500331        0.0000000000
5          148.4131591026       148.4131591026       0.0000000000
6          403.4287934927       403.4287934927       0.0000000000
7          1096.6331584285      1096.6331584285      0.0000000000
8          2980.9579870417      2980.9579870417      0.0000000000
9          8103.0839275754      8103.0839275754      0.0000000000
10         22026.4657948067     22026.4657948067     0.0000000000

Содержимое файла exp_function.txt:
11
0.0 1.0
1.0 2.718281828459045
2.0 7.38905609893065
3.0 20.085536923187668
4.0 54.598150033144236
5.0 148.4131591025766
6.0 403.4287934927351
7.0 1096.6331584284585
8.0 2980.9579870417283
9.0 8103.083927575384
10.0 22026.465794806718
```

```
5. Тестирование логарифма с записью в файл:
==========================================

Бинарная функция записана в файл: ln_function.bin

Сравнение исходного и считанного логарифма:
x          исходный ln(x)       считанный            разница
-----------------------------------------------------------------
1          0.0000000000         0.0000000000         0.0000000000
2          0.6931471806         0.6931471806         0.0000000000
3          1.0986122887         1.0986122887         0.0000000000
4          1.3862943611         1.3862943611         0.0000000000
5          1.6094379124         1.6094379124         0.0000000000
6          1.7917594692         1.7917594692         0.0000000000
7          1.9459101491         1.9459101491         0.0000000000
8          2.0794415417         2.0794415417         0.0000000000
9          2.1972245773         2.1972245773         0.0000000000
10         2.3025850930         2.3025850930         0.0000000000

Размер бинарного файла: 164 байт
Первые 50 байт файла в HEX:
00 00 00 0A 3F F0 00 00 00 00 00 00 00 00 00 00
00 00 00 00 40 00 00 00 00 00 00 00 3F E6 2E 42
FE FA 39 EF 40 08 00 00 00 00 00 00 3F F1 93 EA
7A AD
```

```
6.1. Сериализация с использованием Serializable:
------------------------------------------------
Создана функция ln(exp(x)) = x (теоретически):
x=0,0, значение=0,0000000000 (теория: 0,0)
x=1,0, значение=1,0000000000 (теория: 1,0)
x=2,0, значение=2,0000000000 (теория: 2,0)
x=3,0, значение=3,0000000000 (теория: 3,0)
x=4,0, значение=4,0000000000 (теория: 4,0)
x=5,0, значение=5,0000000000 (теория: 5,0)
x=6,0, значение=6,0000000000 (теория: 6,0)
x=7,0, значение=7,0000000000 (теория: 7,0)
x=8,0, значение=8,0000000000 (теория: 8,0)
x=9,0, значение=9,0000000000 (теория: 9,0)
x=10,0, значение=10,0000000000 (теория: 10,0)

Объект ArrayTabulatedFunction сериализован в файл: function_serializable.ser
Размер файла: 435 байт
```

```
Сравнение исходной и десериализованной функции (ln(exp(x))):
x          исходная             десериализ.          разница
------------------------------------------------------------------
0          0.0000000000         0.0000000000         0.0000000000
1          1.0000000000         1.0000000000         0.0000000000
2          2.0000000000         2.0000000000         0.0000000000
3          3.0000000000         3.0000000000         0.0000000000
4          4.0000000000         4.0000000000         0.0000000000
5          5.0000000000         5.0000000000         0.0000000000
6          6.0000000000         6.0000000000         0.0000000000
7          7.0000000000         7.0000000000         0.0000000000
8          8.0000000000         8.0000000000         0.0000000000
9          9.0000000000         9.0000000000         0.0000000000
10         10.0000000000        10.0000000000        0.0000000000
Максимальная разница: 0,0000000000

Первые 50 байт файла Serializable в HEX:
AC ED 00 05 73 72 00 20 66 75 6E 63 74 69 6F 6E
73 2E 41 72 72 61 79 54 61 62 75 6C 61 74 65 64
46 75 6E 63 74 69 6F 6E 00 00 00 00 00 00 00 02
02 00
```

```
6.2. Сериализация с использованием Externalizable:
-------------------------------------------------
Объект ArrayTabulatedFunctionExternalizable сериализован в файл: function_externalizable.ext
Размер файла: 250 байт

Сравнение исходной и десериализованной функции Externalizable:
x          исходная             десериализ.          разница
------------------------------------------------------------------
0          0.0000000000         0.0000000000         0.0000000000
1          1.0000000000         1.0000000000         0.0000000000
2          2.0000000000         2.0000000000         0.0000000000
3          3.0000000000         3.0000000000         0.0000000000
4          4.0000000000         4.0000000000         0.0000000000
5          5.0000000000         5.0000000000         0.0000000000
6          6.0000000000         6.0000000000         0.0000000000
7          7.0000000000         7.0000000000         0.0000000000
8          8.0000000000         8.0000000000         0.0000000000
9          9.0000000000         9.0000000000         0.0000000000
10         10.0000000000        10.0000000000        0.0000000000
Максимальная разница: 0,0000000000
```