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

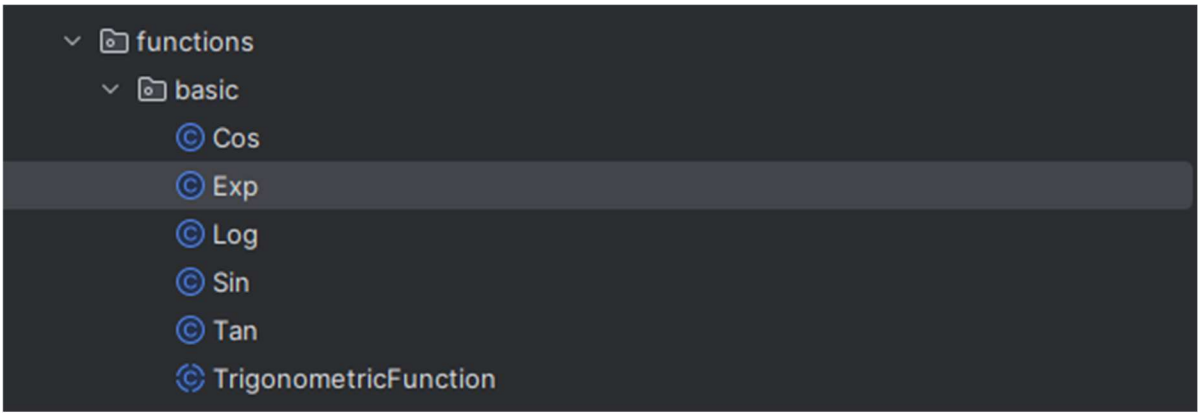
Задание 2.

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.

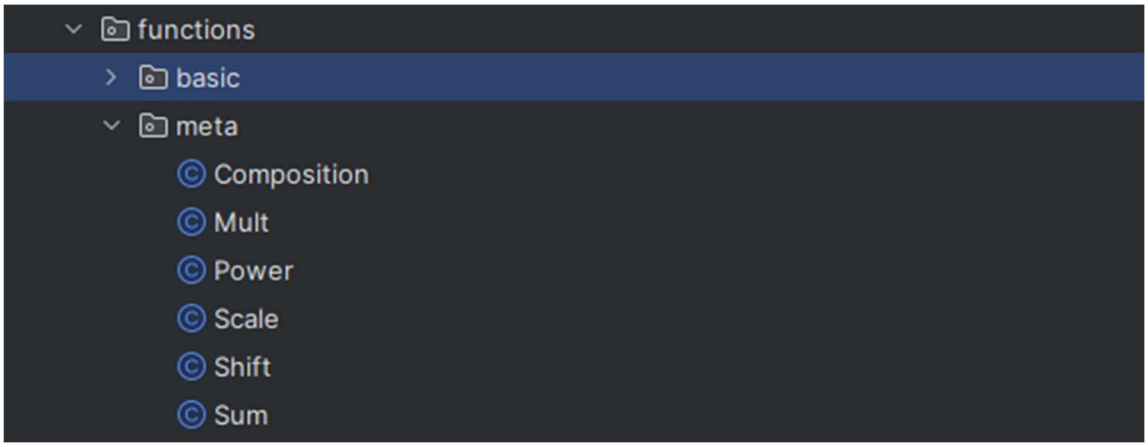
Создайте пакет functions.basic, в нём будут описаны классы ряда функций, заданных аналитически.



```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
}
```

Задание 8.

Проверьте работу написанных классов.

```
1. Табулирование функции sin(x) на [0, π]:
Табулированная функция:
----------------------
Точка 0: (0,0000, 0,0000)
Точка 1: (0,7854, 0,7071)
Точка 2: (1,5708, 1,0000)
Точка 3: (2,3562, 0,7071)
Точка 4: (3,1416, 0,0000)
----------------------
Область определения: [0,0000, 3,1416]
Количество точек: 5

2. Проверка обработки ошибок:
   ОШИБКА: Должно было быть выброшено исключение!
   ✓ Правильно выброшено исключение: Количество точек должно быть не менее 2. Получено: 1

3. Создание функции из массива точек:
Табулированная функция:
----------------------
Точка 0: (0,0000, 0,0000)
Точка 1: (1,0000, 1,0000)
Точка 2: (2,0000, 4,0000)
Точка 3: (3,0000, 9,0000)
----------------------
Область определения: [0,0000, 3,0000]
Количество точек: 4

4. Интерполяция значения в точке x = 1.5:
   f(1.5) ≈ 2.5 (ожидается 2.25)
```

```
5. Численное дифференцирование:
   f'(1.0) ≈ 1.9999999999999445 (ожидается около 2.0)

6. Численное интегрирование методом трапеций:
   ∫₀² x² dx ≈ 0.5 (ожидается около 2.6666666666666665 ≈ 2.6667)

7. Табулирование exp(x) на полной области определения:
Табулированная функция:
----------------------
Точка 0: (-1,0000, 0,3679)
Точка 1: (-0,5000, 0,6065)
Точка 2: (0,0000, 1,0000)
Точка 3: (0,5000, 1,6487)
Точка 4: (1,0000, 2,7183)
----------------------
Область определения: [-1,0000, 1,0000]
Количество точек: 5

8. Создание функции из массивов координат:
Табулированная функция:
----------------------
Точка 0: (0,0000, 0,0000)
Точка 1: (0,5000, 0,2500)
Точка 2: (1,0000, 1,0000)
Точка 3: (1,5000, 2,2500)
Точка 4: (2,0000, 4,0000)
----------------------
Область определения: [0,0000, 2,0000]
Количество точек: 5
```

```
9. Тестирование методов ввода/вывода:
   Байтовый вывод: записано 68 байт
   Байтовый ввод: функция восстановлена
   Символьный вывод: записано 39 символов
   Символьный ввод: функция восстановлена
   Байтовое восстановление корректно: ДА
   Символьное восстановление корректно: ДА

10. Попытка создания объекта класса TabulatedFunctions:
   ✓ Нельзя создать объект класса TabulatedFunctions: Нельзя создавать объекты утилитного класса TabulatedFunctions
```

Задание 9.

Сделайте классы сериализуемыми (ответ в задании 8.)