Лабораторная работа №4

Коновалов Сергей Сергеевич 6204-010302D

Содержание

Задание 1

Добавил конструкторы, принимающие массив объектов FunctionPoint, в оба класса табулированных функций.

```
public ArrayTabulatedFunction(FunctionPoint[] points) {
    if (points.length < 2) {
        throw new IllegalArgumentException("Количество точек не может быть меньше двух");
    }

    // Проверка упорядоченности точек по x
    for (int i = 0; i < points.length - 1; i++) {
        if (points[i].getX() >= points[i + 1].getX()) {
            throw new IllegalArgumentException("Точки должны быть упорядочены по возрастанию x");
        }
    }

    this.pointsCount = points.length;
    this.points = new FunctionPoint[pointsCount];

    // Создаем копии точек для обеспечения инкапсуляции
    for (int i = 0; i < pointsCount; i++) {
        this.points[i] = new FunctionPoint(points[i]);
    }
}
public LinkedListTabulatedFunction(FunctionPoint[] points) {
    if (points.length < 2) {
        throw new IllegalArgumentException("Количество точек не может быть меньше двух");
    }

    // Проверка упорядоченности точек по x
    for (int i = 0; i < points.length - 1; i++) {
        if (points[i].getX() >= points[i + 1].getX()) {
            throw new IllegalArgumentException("Точки должны быть упорядочены по возрастанию x");
        }
    }

    initializeList();

    // Добавляем точки в список, создавая копии для инкапсуляции
    for (int i = 0; i < points.length; i++) {
        addNodeToTail().setPoint(new FunctionPoint(points[i]));
    }
}
```

Задание 2

Внёс необходимые изменения для создания интерфейса Function и рефакторинга TabulatedFunction

```
package functions;

public interface Function {

    double getLeftDomainBorder();

    double getRightDomainBorder();

    double getFunctionValue(double x);
}
```

```
package functions;

public interface TabulatedFunction extends Function {

    int getPointsCount();

    FunctionPoint getPoint(int index);

    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;

    double getPointX(int index);

    double getPointY(int index);

    void setPointX(int index, double x) throws InappropriateFunctionPointException;

    void setPointY(int index, double y);

    void deletePoint(int index);

    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
}
```

Задание 3

Создал пакет functions.basic и создал классы Exp.java, Log.java, TrigonometricFunction.java, Sin.java, Cos.java, Tan.java:

```java
package functions.basic;

import functions.Function;

public class Exp implements Function {

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

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

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

```java
package functions.basic;

import functions.Function;

public class Log implements Function {
    private double base;

    public Log(double base) {
        if (base <= 0 || base == 1) {
            throw new IllegalArgumentException("Основание логарифма должно быть положительным и не равным 1");
        }
        this.base = base;
    }

    public double getLeftDomainBorder() {
        return 0.0; // Логарифм определен для x > 0
    }

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

    public double getFunctionValue(double x) {
        if (x <= 0) {
            return Double.NaN; // Логарифм не определен для неположительных x
        }
        return Math.log(x) / Math.log(base); // Формула смены основания
    }

    public double getBase() {
        return base;
    }
}
```

```java
package functions.basic;

import functions.Function;

public abstract class TrigonometricFunction implements Function {

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

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

    // Абстрактный метод, который должны реализовать наследники
    public abstract double getFunctionValue(double x);
}
```

```java
package functions.basic;

public class Sin extends TrigonometricFunction {

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

```java
package functions.basic;

public class Cos extends TrigonometricFunction {

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

```java
package functions.basic;

public class Tan extends TrigonometricFunction {

    public double getFunctionValue(double x) {
        // Тангенс не определен, когда косинус равен 0
        double cosValue = Math.cos(x);
        if (Math.abs(cosValue) < 1e-10) {
            return Double.NaN;
        }
        return Math.tan(x);
    }
}
```

Задание 4

Создал пакет functions.meta и создал классы Sum.java, Mult.java, Power.java, Scale.java, Shift.java, Composition.java:

```java
package functions.meta;

import functions.Function;

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) {
        // Проверяем, что x принадлежит пересечению областей определения
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        return f1.getFunctionValue(x) + f2.getFunctionValue(x);
    }
}
```

```java
package functions.meta;

import functions.Function;

public class Mult implements Function {
    private Function f1;
    private Function f2;

    public Mult(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) {
        // Проверяем, что x принадлежит пересечению областей определения
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        return f1.getFunctionValue(x) * f2.getFunctionValue(x);
    }
}
```

```java
package functions.meta;

import functions.Function;

public class Power implements Function {
    private Function f;
    private double power;

    public Power(Function f, double power) {
        this.f = f;
        this.power = power;
    }

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

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

    public double getFunctionValue(double x) {
        double baseValue = f.getFunctionValue(x);
        if (Double.isNaN(baseValue)) {
            return Double.NaN;
        }
        return Math.pow(baseValue, power);
    }
}
```

```java
package functions.meta;

import functions.Function;

public class Scale implements Function {
    private Function f;
    private double scaleX;
    private double scaleY;

    public Scale(Function f, double scaleX, double scaleY) {
        this.f = f;
        this.scaleX = scaleX;
        this.scaleY = scaleY;
    }

    public double getLeftDomainBorder() {
        if (scaleX > 0) {
            return f.getLeftDomainBorder() / scaleX;
        } else if (scaleX < 0) {
            return f.getRightDomainBorder() / scaleX;
        } else {
            return Double.NaN; // При scaleX = 0 функция вырождена
        }
    }

    public double getRightDomainBorder() {
        if (scaleX > 0) {
            return f.getRightDomainBorder() / scaleX;
        } else if (scaleX < 0) {
            return f.getLeftDomainBorder() / scaleX;
        } else {
            return Double.NaN; // При scaleX = 0 функция вырождена
        }
    }

    public double getFunctionValue(double x) {
        double scaledX = x * scaleX;
        // Проверяем, что scaledX принадлежит области определения исходной функции
        if (scaledX < f.getLeftDomainBorder() || scaledX > f.getRightDomainBorder()) {
            return Double.NaN;
        }
        return f.getFunctionValue(scaledX) * scaleY;
    }
}
```

```java
package functions.meta;

import functions.Function;

public class Shift implements Function {
    private Function f;
    private double shiftX;
    private double shiftY;

    public Shift(Function f, double shiftX, double shiftY) {
        this.f = f;
        this.shiftX = shiftX;
        this.shiftY = shiftY;
    }

    public double getLeftDomainBorder() {
        return f.getLeftDomainBorder() - shiftX;
    }

    public double getRightDomainBorder() {
        return f.getRightDomainBorder() - shiftX;
    }

    public double getFunctionValue(double x) {
        double shiftedX = x + shiftX;
        // Проверяем, что shiftedX принадлежит области определения исходной функции
        if (shiftedX < f.getLeftDomainBorder() || shiftedX > f.getRightDomainBorder()) {
            return Double.NaN;
        }
        return f.getFunctionValue(shiftedX) + shiftY;
    }
}
```

```java
package functions.meta;

import functions.Function;

public class Composition implements Function {
    private Function f1;
    private Function f2;

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

    public double getLeftDomainBorder() {
        return f1.getLeftDomainBorder();
    }

    public double getRightDomainBorder() {
        return f1.getRightDomainBorder();
    }

    public double getFunctionValue(double x) {
        double innerValue = f1.getFunctionValue(x);
        if (Double.isNaN(innerValue)) {
            return Double.NaN;
        }
        return f2.getFunctionValue(innerValue);
    }
}
```

Задание 5

Создал класс Functions со статическими методами-фабриками для работы с функциями

```java
package functions;

import functions.meta.*;

public class Functions {

    private Functions() {
        throw new AssertionError("Нельзя создать объект класса Functions");
    }

    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);
    }
}
```

Задание 6

Создал класс TabulatedFunctions со статическим методом tabulate для табулирования функций

```java
package functions;

public class TabulatedFunctions {

    private TabulatedFunctions() {
        throw new AssertionError("Нельзя создать объект класса TabulatedFunctions");
    }

    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
        if (pointsCount < 2) {
            throw new IllegalArgumentException("Количество точек табулирования не может быть меньше 2");
        }

        if (leftX >= rightX) {
            throw new IllegalArgumentException("Левая граница табулирования должна быть меньше правой");
        }

        if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
            throw new IllegalArgumentException(
                "Границы табулирования [" + leftX + ", " + rightX + "] " +
                "выходят за область определения функции [" +
                function.getLeftDomainBorder() + ", " + function.getRightDomainBorder() + "]"
            );
        }

        double[] values = new double[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);

        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            values[i] = function.getFunctionValue(x);
        }

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

    public static TabulatedFunction tabulate(Function function, double leftX, double rightX) {
        return tabulate(function, leftX, rightX, 20); // По умолчанию 20 точек
    }
}
```

Задание 7

Добавил методы ввода/вывода в класс TabulatedFunctions

```java
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) {
    DataOutputStream dataOut = new DataOutputStream(out);
    try {
        dataOut.writeInt(function.getPointsCount());
        for (int i = 0; i < function.getPointsCount(); i++) {
            dataOut.writeDouble(function.getPointX(i));
            dataOut.writeDouble(function.getPointY(i));
        }
        dataOut.flush();
    } catch (IOException e) {
        throw new RuntimeException("Ошибка при выводе функции в поток", e);
    }
}
```

```
public static TabulatedFunction inputTabulatedFunction(InputStream in) {
    DataInputStream dataIn = new DataInputStream(in);
    try {
        int pointsCount = dataIn.readInt();
        double[] xValues = new double[pointsCount];
        double[] yValues = new double[pointsCount];

        for (int i = 0; i < pointsCount; i++) {
            xValues[i] = dataIn.readDouble();
            yValues[i] = dataIn.readDouble();
        }

        return new ArrayTabulatedFunction(xValues, yValues);
    } catch (IOException e) {
        throw new RuntimeException("Ошибка при чтении функции из потока", e);
    }
}

public static void writeTabulatedFunction(TabulatedFunction function, Writer out) {
    PrintWriter writer = new PrintWriter(out);
    try {
        writer.print(function.getPointsCount());
        writer.print(' ');

        for (int i = 0; i < function.getPointsCount(); i++) {
            writer.print(function.getPointX(i));
            writer.print(' ');
            writer.print(function.getPointY(i));
            if (i < function.getPointsCount() - 1) {
                writer.print(' ');
            }
        }
        writer.flush();
    } catch (Exception e) {
        throw new RuntimeException("Ошибка при записи функции в символьный поток", e);
    }
}
public static TabulatedFunction readTabulatedFunctionAlt(Reader in) {
    BufferedReader reader = new BufferedReader(in);
    try {
        String line = reader.readLine();
        if (line == null) {
            throw new RuntimeException("Пустой поток");
        }

        StringTokenizer tokenizer = new StringTokenizer(line);

        if (!tokenizer.hasMoreTokens()) {
            throw new RuntimeException("Ожидалось количество точек");
        }

        int pointsCount = Integer.parseInt(tokenizer.nextToken());

        if (pointsCount < 2) {
            throw new RuntimeException("Некорректное количество точек: " + pointsCount);
        }

        double[] xValues = new double[pointsCount];
        double[] yValues = new double[pointsCount];

        for (int i = 0; i < pointsCount; i++) {
            if (!tokenizer.hasMoreTokens()) {
                throw new RuntimeException("Ожидалась x-координата точки " + i);
            }
            xValues[i] = Double.parseDouble(tokenizer.nextToken());

            if (!tokenizer.hasMoreTokens()) {
                throw new RuntimeException("Ожидалась y-координата точки " + i);
            }
            yValues[i] = Double.parseDouble(tokenizer.nextToken());
        }

        return new ArrayTabulatedFunction(xValues, yValues);

    } catch (IOException e) {
        throw new RuntimeException("Ошибка при чтении табулированной функции из символьного потока", e);
    }
}
```

Задание 8

```java
private static void test() {
    System.out.println("-".repeat(80));
    System.out.println("Тестирование работы всех написанных классов");
    System.out.println("-".repeat(80));

    testBasicFunctionsSinCos();
    testTabulatedAnalogues();
    testSumOfSquares();
    testFileOperationsExponential();
    testFileOperationsLogarithm();
    compareStorageFormats();
}

private static void testBasicFunctionsSinCos() {
    System.out.println("\n1. Тестирование базовых функций Sin и Cos");
    System.out.println("-".repeat(50));

    Function sin = new Sin();
    Function cos = new Cos();

    double from = 0;
    double to = Math.PI;
    double step = 0.1;

    System.out.println("Значения sin(x) на [0, π] с шагом 0.1:");
    System.out.printf("%-8s %-10s%n", "x", "sin(x)");
    for (double x = from; x <= to + 1e-10; x += step) {
        System.out.printf("%-8.3f %-10.6f%n", x, sin.getFunctionValue(x));
    }

    System.out.println("\nЗначения cos(x) на [0, π] с шагом 0.1:");
    System.out.printf("%-8s %-10s%n", "x", "cos(x)");
    for (double x = from; x <= to + 1e-10; x += step) {
        System.out.printf("%-8.3f %-10.6f%n", x, cos.getFunctionValue(x));
    }
}

private static void testTabulatedAnalogues() {
    System.out.println("\n\n2. Табулированные аналоги Sin и Cos (10 точек)");
    System.out.println("-".repeat(55));

    Function sin = new Sin();
    Function cos = new Cos();

    TabulatedFunction tabulatedSin = TabulatedFunctions.tabulate(sin, 0, Math.PI, 10);
    TabulatedFunction tabulatedCos = TabulatedFunctions.tabulate(cos, 0, Math.PI, 10);

    double from = 0;
    double to = Math.PI;
    double step = 0.1;

    System.out.println("Сравнение sin(x) и табулированного sin(x) (10 точек):");
    System.out.printf("%-8s %-12s %-12s %-12s%n", "x", "sin(x)", "tab_sin(x)", "погрешность");
    for (double x = from; x <= to + 1e-10; x += step) {
        double exact = sin.getFunctionValue(x);
        double approx = tabulatedSin.getFunctionValue(x);
        double error = Math.abs(exact - approx);
        System.out.printf("%-8.3f %-12.6f %-12.6f %-12.6f%n", x, exact, approx, error);
    }

    System.out.println("\nСравнение cos(x) и табулированного cos(x) (10 точек):");
    System.out.printf("%-8s %-12s %-12s %-12s%n", "x", "cos(x)", "tab_cos(x)", "погрешность");
    for (double x = from; x <= to + 1e-10; x += step) {
        double exact = cos.getFunctionValue(x);
        double approx = tabulatedCos.getFunctionValue(x);
        double error = Math.abs(exact - approx);
        System.out.printf("%-8.3f %-12.6f %-12.6f %-12.6f%n", x, exact, approx, error);
    }
}

private static void testSumOfSquares() {
    System.out.println("\n\n3. Сумма квадратов табулированных функций");
    System.out.println("-".repeat(45));

    int[] pointsCounts = {5, 10, 20};

    for (int pointsCount : pointsCounts) {
        System.out.println("\nКоличество точек в табулированных функциях: " + pointsCount);

        TabulatedFunction tabulatedSin = TabulatedFunctions.tabulate(new Sin(), 0, Math.PI, pointsCount);
        TabulatedFunction tabulatedCos = TabulatedFunctions.tabulate(new Cos(), 0, Math.PI, pointsCount);

        Function sumOfSquares = Functions.sum(
            Functions.power(tabulatedSin, 2),
            Functions.power(tabulatedCos, 2)
        );

        double from = 0;
        double to = Math.PI;
        double step = 0.1;

        System.out.printf("%-8s %-15s%n", "x", "sin²(x)+cos²(x)");
        for (double x = from; x <= to + 1e-10; x += step) {
            double value = sumOfSquares.getFunctionValue(x);
            double deviation = Math.abs(value - 1.0);
            System.out.printf("%-8.3f %-15.8f (отклонение: %.8f)%n", x, value, deviation);
        }
    }
}
```

```
private static void testFileOperationsExponential() {
    System.out.println("\n\n4. Работа с текстовыми файлами (экспонента)");
    System.out.println("-".repeat(50));

    String filename = "exponential_function.txt";

    try {
        TabulatedFunction expFunction = TabulatedFunctions.tabulate(new Exp(), 0, 10, 11);

        try (FileWriter writer = new FileWriter(filename)) {
            TabulatedFunctions.writeTabulatedFunction(expFunction, writer);
        }
        System.out.println("Табулированная экспонента записана в файл: " + filename);

        TabulatedFunction readFunction;
        try (FileReader reader = new FileReader(filename)) {
            readFunction = TabulatedFunctions.readTabulatedFunction(reader);
        }
        System.out.println("Функция прочитана из текстового файла");

        System.out.println("\nСравнение исходной и прочитанной функции:");
        System.out.printf("%-8s %-15s %-15s %-15s%n", "x", "исходная", "прочитанная", "разница");
        for (int i = 0; i < expFunction.getPointsCount(); i++) {
            double x = expFunction.getPointX(i);
            double original = expFunction.getPointY(i);
            double read = readFunction.getPointY(i);
            double difference = Math.abs(original - read);
            System.out.printf("%-8.1f %-15.8f %-15.8f %-15.8f%n", x, original, read, difference);
        }

        System.out.println("\nСодержимое текстового файла:");
        String content = new String(Files.readAllBytes(Paths.get(filename)));
        System.out.println(content);

        Files.deleteIfExists(Paths.get(filename));
        System.out.println("Временный файл удален");

    } catch (Exception e) {
        System.out.println("Ошибка: " + e.getMessage());
    }
}

private static void testFileOperationsLogarithm() {
    System.out.println("\n\n5. Работа с бинарными файлами (логарифм)");
    System.out.println("-".repeat(50));

    String filename = "logarithm_function.dat";

    try {
        TabulatedFunction logFunction = TabulatedFunctions.tabulate(new Log(Math.E), 1, 10, 11);

        try (FileOutputStream out = new FileOutputStream(filename)) {
            TabulatedFunctions.outputTabulatedFunction(logFunction, out);
        }
        System.out.println("Табулированный логарифм записан в файл: " + filename);

        TabulatedFunction readFunction;
        try (FileInputStream in = new FileInputStream(filename)) {
            readFunction = TabulatedFunctions.inputTabulatedFunction(in);
        }
        System.out.println("Функция прочитана из бинарного файла");

        System.out.println("\nСравнение исходной и прочитанной функции:");
        System.out.printf("%-8s %-15s %-15s %-15s%n", "x", "исходная", "прочитанная", "разница");
        for (int i = 0; i < logFunction.getPointsCount(); i++) {
            double x = logFunction.getPointX(i);
            double original = logFunction.getPointY(i);
            double read = readFunction.getPointY(i);
            double difference = Math.abs(original - read);
            System.out.printf("%-8.1f %-15.8f %-15.8f %-15.8f%n", x, original, read, difference);
        }

        File file = new File(filename);
        System.out.println("\nРазмер бинарного файла: " + file.length() + " байт");

        Files.deleteIfExists(Paths.get(filename));
        System.out.println("Временный файл удален");

    } catch (Exception e) {
        System.out.println("Ошибка: " + e.getMessage());
    }
}
```

```
private static void compareStorageFormats() {
    System.out.println("\n\n6. Сравнение орматов хранения");
    System.out.println("-".repeat(35));

    String textFile = "comparison_text.txt";
    String binaryFile = "comparison_binary.dat";

    try {
        TabulatedFunction testFunction = TabulatedFunctions.tabulate(new Sin(), 0, Math.PI, 5);

        try (FileWriter writer = new FileWriter(textFile)) {
            TabulatedFunctions.writeTabulatedFunction(testFunction, writer);
        }

        try (FileOutputStream out = new FileOutputStream(binaryFile)) {
            TabulatedFunctions.outputTabulatedFunction(testFunction, out);
        }

        File text = new File(textFile);
        File binary = new File(binaryFile);

        System.out.println("Размер текстового файла: " + text.length() + " байт");
        System.out.println("Размер бинарного файла: " + binary.length() + " байт");
        System.out.println("Бинарный файл занимает " +
            String.format("%.1f", (double)binary.length() / text.length() * 100) + "% от текстового");

        System.out.println("\nСодержимое текстового файла:");
        System.out.println(new String(Files.readAllBytes(Paths.get(textFile))));

        Files.deleteIfExists(Paths.get(textFile));
        Files.deleteIfExists(Paths.get(binaryFile));
        System.out.println("\nВременные файлы удалены");

    } catch (Exception e) {
        System.out.println("Ошибка: " + e.getMessage());
    }
}

public static void main(String[] args) {
    test();
}
```

Задание 9

Добавил использование Serializable и Externalizable в классах ArrayTabulatedFunction и LinkedListTabulatedFunction, а также сделал отдельные функции, используемые для Externalizable:

```
@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 {
    pointsCount = in.readInt();
    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);
    }
}
```

```
@Override
public void writeExternal(ObjectOutput out) throws IOException {
    out.writeInt(pointsCount);
    FunctionNode current = head.getNext();
    while (current != head) {
        out.writeDouble(current.getPoint().getX());
        out.writeDouble(current.getPoint().getY());
        current = current.getNext();
    }
}

@Override
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
    initializeList();
    int count = in.readInt();
    for (int i = 0; i < count; i++) {
        double x = in.readDouble();
        double y = in.readDouble();
        addNodeToTail().setPoint(new FunctionPoint(x, y));
    }
}
```

Вывод Main:

```
PS D:\Other\Study\GitHubLab4> javac functions/*.java
PS D:\Other\Study\GitHubLab4> javac Main.java
PS D:\Other\Study\GitHubLab4> java Main
--------------------------------------------------------------------------------
Тестирование работы всех написанных классов
--------------------------------------------------------------------------------

1. Тестирование базовых функций Sin и Cos
------------------------------------------------
Значения sin(x) на [0, ?] с шагом 0.1:
x        sin(x)
0,000    0,000000
0,100    0,099833
0,200    0,198669
0,300    0,295520
0,400    0,389418
0,500    0,479426
0,600    0,564642
0,700    0,644218
0,800    0,717356
0,900    0,783327
1,000    0,841471
1,100    0,891207
1,200    0,932039
1,300    0,963558
1,400    0,985450
1,500    0,997495
1,600    0,999574
1,700    0,991665
1,800    0,973848
1,900    0,946300
2,000    0,909297
2,100    0,863209
2,200    0,808496
2,300    0,745705
2,400    0,675463
2,500    0,598472
2,600    0,515501
2,700    0,427380
2,800    0,334988
2,900    0,239249
3,000    0,141120
3,100    0,041581

Значения cos(x) на [0, ?] с шагом 0.1:
x        cos(x)
0,000    1,000000
0,100    0,995004
0,200    0,980067
0,300    0,955336
0,400    0,921061
0,500    0,877583
0,600    0,825336
0,700    0,764842
0,800    0,696707
0,900    0,621610
1,000    0,540302
1,100    0,453596
1,200    0,362358
1,300    0,267499
1,400    0,169967
1,500    0,070737
1,600    -0,029200
1,700    -0,128844
1,800    -0,227202
1,900    -0,323290
2,000    -0,416147
2,100    -0,504846
2,200    -0,588501
2,300    -0,666276
2,400    -0,737394
2,500    -0,801144
2,600    -0,856889
2,700    -0,904072
2,800    -0,942222
2,900    -0,970958
3,000    -0,989992
3,100    -0,999135
```

```
2. Табулированные аналоги Sin и Cos (10 точек)
----------------------------------------------------------
Сравнение sin(x) и табулированного sin(x) (10 точек):
x        sin(x)       tab_sin(x)   погрешность
0,000    0,000000     0,000000     0,000000
0,100    0,099833     0,097982     0,001852
0,200    0,198669     0,195963     0,002706
0,300    0,295520     0,293945     0,001576
0,400    0,389418     0,385907     0,003512
0,500    0,479426     0,472070     0,007355
0,600    0,564642     0,558234     0,006409
0,700    0,644218     0,643982     0,000235
0,800    0,717356     0,707935     0,009421
0,900    0,783327     0,771888     0,011439
1,000    0,841471     0,835841     0,005630
1,100    0,891207     0,883993     0,007214
1,200    0,932039     0,918022     0,014017
1,300    0,963558     0,952051     0,011508
1,400    0,985450     0,984808     0,000642
1,500    0,997495     0,984808     0,012687
1,600    0,999574     0,984808     0,014766
1,700    0,991665     0,984808     0,006857
1,800    0,973848     0,966204     0,007644
1,900    0,946300     0,932175     0,014125
2,000    0,909297     0,898147     0,011151
2,100    0,863209     0,862441     0,000768
2,200    0,808496     0,798488     0,010008
2,300    0,745705     0,734535     0,011170
2,400    0,675463     0,670582     0,004881
2,500    0,598472     0,594072     0,004401
2,600    0,515501     0,507908     0,007593
2,700    0,427380     0,421745     0,005635
2,800    0,334988     0,334698     0,000290
2,900    0,239249     0,236716     0,002533
3,000    0,141120     0,138735     0,002385
3,100    0,041581     0,040753     0,000828

Сравнение cos(x) и табулированного cos(x) (10 точек):
x        cos(x)       tab_cos(x)   погрешность
0,000    1,000000     1,000000     0,000000
0,100    0,995004     0,982723     0,012281
0,200    0,980067     0,965446     0,014620
0,300    0,955336     0,948170     0,007167
0,400    0,921061     0,914355     0,006706
0,500    0,877583     0,864608     0,012974
0,600    0,825336     0,814862     0,010474
0,700    0,764842     0,764620     0,000222
0,800    0,696707     0,688404     0,008302
0,900    0,621610     0,612188     0,009422
1,000    0,540302     0,535972     0,004330
1,100    0,453596     0,450633     0,002963
1,200    0,362358     0,357141     0,005217
1,300    0,267499     0,263648     0,003851
1,400    0,169967     0,169931     0,000037
1,500    0,070737     0,070437     0,000300
1,600    -0,029200    -0,029056    0,000144
1,700    -0,128844    -0,128549    0,000296
1,800    -0,227202    -0,224761    0,002441
1,900    -0,323290    -0,318254    0,005035
2,000    -0,416147    -0,411747    0,004400
2,100    -0,504846    -0,504272    0,000574
2,200    -0,588501    -0,580488    0,008013
2,300    -0,666276    -0,656704    0,009572
2,400    -0,737394    -0,732920    0,004474
2,500    -0,801144    -0,794171    0,006973
2,600    -0,856889    -0,843917    0,012972
2,700    -0,904072    -0,893664    0,010408
2,800    -0,942222    -0,940984    0,001239
2,900    -0,970958    -0,958261    0,012698
3,000    -0,989992    -0,975537    0,014455
3,100    -0,999135    -0,992814    0,006321
```

```
3. Сумма квадратов табулированных функций
-----------------------------------------

Количество точек в табулированных функциях: 5
x        sin?(x)+cos?(x)
0,000    1,00000000      (отклонение: 0,00000000)
0,100    0,93491177      (отклонение: 0,06508823)
0,200    0,88881636      (отклонение: 0,11118364)
0,300    0,86171377      (отклонение: 0,13828623)
0,400    0,85360401      (отклонение: 0,14639599)
0,500    0,86448707      (отклонение: 0,13551293)
0,600    0,89436296      (отклонение: 0,10563704)
0,700    0,94323167      (отклонение: 0,05676833)
0,800    0,98931175      (отклонение: 0,01068825)
0,900    0,92699682      (отклонение: 0,07300318)
1,000    0,88367471      (отклонение: 0,11632529)
1,100    0,85934542      (отклонение: 0,14065458)
1,200    0,85400896      (отклонение: 0,14599104)
1,300    0,86766533      (отклонение: 0,13233467)
1,400    0,90031451      (отклонение: 0,09968549)
1,500    0,95195653      (отклонение: 0,04804347)
1,600    0,97902845      (отклонение: 0,02097155)
1,700    0,91948682      (отклонение: 0,08051318)
1,800    0,87893801      (отклонение: 0,12106199)
1,900    0,85738203      (отклонение: 0,14261797)
2,000    0,85481887      (отклонение: 0,14518113)
2,100    0,87124853      (отклонение: 0,12875147)
2,200    0,90667102      (отклонение: 0,09332898)
2,300    0,96108634      (отклонение: 0,03891366)
2,400    0,96915010      (отклонение: 0,03084990)
2,500    0,91238177      (отклонение: 0,08761823)
2,600    0,87460627      (отклонение: 0,12539373)
2,700    0,85582359      (отклонение: 0,14417641)
2,800    0,85603373      (отклонение: 0,14396627)
2,900    0,87523669      (отклонение: 0,12476331)
3,000    0,91343248      (отклонение: 0,08656752)
3,100    0,97062110      (отклонение: 0,02937890)

Количество точек в табулированных функциях: 10
x        sin?(x)+cos?(x)
0,000    1,00000000      (отклонение: 0,00000000)
0,100    0,97534529      (отклонение: 0,02465471)
0,200    0,97048832      (отклонение: 0,02951168)
0,300    0,98542910      (отклонение: 0,01457090)
0,400    0,98496848      (отклонение: 0,01503152)
0,500    0,97039758      (отклонение: 0,02960242)
0,600    0,97562443      (отклонение: 0,02437557)
0,700    0,99935789      (отклонение: 0,00064211)
0,800    0,97507306      (отклонение: 0,02492694)
0,900    0,97058598      (отклонение: 0,02941402)
1,000    0,98589664      (отклонение: 0,01410336)
1,100    0,98451477      (отклонение: 0,01548523)
1,200    0,97031375      (отклонение: 0,02968625)
1,300    0,97591048      (отклонение: 0,02408952)
1,400    0,99872269      (отклонение: 0,00127731)
1,500    0,97480774      (отклонение: 0,02519226)
1,600    0,97069054      (отклонение: 0,02930946)
1,700    0,98637108      (отклонение: 0,01362892)
1,800    0,98406796      (отклонение: 0,01593204)
1,900    0,97023683      (отклонение: 0,02976317)
2,000    0,97620344      (отклонение: 0,02379656)
2,100    0,99809440      (отклонение: 0,00190560)
2,200    0,97454934      (отклонение: 0,02545066)
2,300    0,97080201      (отклонение: 0,02919799)
2,400    0,98685244      (отклонение: 0,01314756)
2,500    0,98362807      (отклонение: 0,01637193)
2,600    0,97016682      (отклонение: 0,02983318)
2,700    0,97650331      (отклонение: 0,02349669)
2,800    0,99747303      (отклонение: 0,00252697)
2,900    0,97429784      (отклонение: 0,02570216)
3,000    0,97092040      (отклонение: 0,02907960)
3,100    0,98734070      (отклонение: 0,01265930)
```

```
Количество точек в табулированных функциях: 20
x        sin?(x)+cos?(x)
0,000    1,00000000      (отклонение: 0,00000000)
0,100    0,99348018      (отклонение: 0,00651982)
0,200    0,99548137      (отклонение: 0,00451863)
0,300    0,99587637      (отклонение: 0,00412363)
0,400    0,99335893      (отклонение: 0,00664107)
0,500    0,99936251      (отклонение: 0,00063749)
0,600    0,99363270      (отклонение: 0,00636730)
0,700    0,99511764      (отклонение: 0,00488236)
0,800    0,99630265      (отклонение: 0,00369735)
0,900    0,99326897      (отклонение: 0,00673103)
1,000    0,99875630      (отклонение: 0,00124370)
1,100    0,99381649      (отклонение: 0,00618351)
1,200    0,99478519      (отклонение: 0,00521481)
1,300    0,99676021      (отклонение: 0,00323979)
1,400    0,99321028      (отклонение: 0,00678972)
1,500    0,99818136      (отклонение: 0,00181864)
1,600    0,99403156      (отклонение: 0,00596844)
1,700    0,99448402      (отклонение: 0,00551598)
1,800    0,99724905      (отклонение: 0,00275095)
1,900    0,99318287      (отклонение: 0,00681713)
2,000    0,99763770      (отклонение: 0,00236230)
2,100    0,99427791      (отклонение: 0,00572209)
2,200    0,99421412      (отклонение: 0,00578588)
2,300    0,99776916      (отклонение: 0,00223084)
2,400    0,99318673      (отклонение: 0,00681327)
2,500    0,99712532      (отклонение: 0,00287468)
2,600    0,99455554      (отклонение: 0,00544446)
2,700    0,99397550      (отклонение: 0,00602450)
2,800    0,99832055      (отклонение: 0,00167945)
2,900    0,99322187      (отклонение: 0,00677813)
3,000    0,99664422      (отклонение: 0,00335578)
3,100    0,99486445      (отклонение: 0,00513555)


4. Работа с текстовыми файлами (экспонента)
--------------------------------------------------
Табулированная экспонента записана в файл: exponential_function.txt
Функция прочитана из текстового файла

Сравнение исходной и прочитанной функции:
x        исходная        прочитанная     разница
0,0      1,00000000      1,00000000      0,00000000
1,0      2,71828183      2,71828183      0,00000000
2,0      7,38905610      7,38905610      0,00000000
3,0      20,08553692     20,08553692     0,00000000
4,0      54,59815003     54,59815003     0,00000000
5,0      148,41315910    148,41315910    0,00000000
6,0      403,42879349    403,42879349    0,00000000
7,0      1096,63315843   1096,63315843   0,00000000
8,0      2980,95798704   2980,95798704   0,00000000
9,0      8103,08392758   8103,08392758   0,00000000
10,0     22026,46579481  22026,46579481  0,00000000

Содержимое текстового файла:
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 6.0 403.42879349273511 7.0 1096.6331584
284585 8.0 2980.9579870417283 9.0 8103.083927575384 10.0 22026.465794806718
Временный файл удален


5. Работа с бинарными файлами (логарифм)
--------------------------------------------------
Табулированный логарифм записан в файл: logarithm_function.dat
Функция прочитана из бинарного файла

Сравнение исходной и прочитанной функции:
x        исходная        прочитанная     разница
1,0      0,00000000      0,00000000      0,00000000
1,9      0,64185389      0,64185389      0,00000000
2,8      1,02961942      1,02961942      0,00000000
3,7      1,30833282      1,30833282      0,00000000
4,6      1,52605630      1,52605630      0,00000000
5,5      1,70474809      1,70474809      0,00000000
6,4      1,85629799      1,85629799      0,00000000
7,3      1,98787435      1,98787435      0,00000000
8,2      2,10413415      2,10413415      0,00000000
9,1      2,20827441      2,20827441      0,00000000
10,0     2,30258509      2,30258509      0,00000000

Размер бинарного файла: 180 байт
Временный файл удален
```

```
6. Сравнение орматов хранения
----------------------------------
Размер текстового файла: 148 байт
Размер бинарного файла: 84 байт
Бинарный файл занимает 56,8% от текстового

Содержимое текстового файла:
5 0.0 0.0 0.7853981633974483 0.7071067811865475 1.5707963267948966 1.0 2.3561944
90192345 0.7071067811865476 3.141592653589793 1.2246467991473532E-16

Временные файлы удалены
================================================================================
ЗАДАНИЕ 9: ТЕСТИРОВАНИЕ СЕРИАЛИЗАЦИИ
================================================================================

1. Сериализация через Serializable
==========================================
Исходная функция создана:
  Тип: ArrayTabulatedFunction
  Количество точек: 11
  Область определения: [0.0, 10.0]
Функция сериализована в файл: serializable_log_exp.ser
Функция десериализована из файла
  Тип после десериализации: ArrayTabulatedFunction

Сравнение исходной и десериализованной функции:
x         исходная        десериал.       разница
0,0       0,00000000      0,00000000      0,00000000
1,0       1,00000000      1,00000000      0,00000000
2,0       2,00000000      2,00000000      0,00000000
3,0       3,00000000      3,00000000      0,00000000
4,0       4,00000000      4,00000000      0,00000000
5,0       5,00000000      5,00000000      0,00000000
6,0       6,00000000      6,00000000      0,00000000
7,0       7,00000000      7,00000000      0,00000000
8,0       8,00000000      8,00000000      0,00000000
9,0       9,00000000      9,00000000      0,00000000
10,0      10,00000000     10,00000000     0,00000000

Размер файла Serializable: 236 байт
Временный файл удален


2. Сериализация через Externalizable
================================================
Исходная функция создана:
  Тип: ArrayTabulatedFunction
  Количество точек: 11
  Область определения: [0.0, 10.0]
Функция сериализована в файл: externalizable_log_exp.ser
Функция десериализована из файла
  Тип после десериализации: ArrayTabulatedFunction

Сравнение исходной и десериализованной функции:
x         исходная        десериал.       разница
0,0       0,00000000      0,00000000      0,00000000
1,0       1,00000000      1,00000000      0,00000000
2,0       2,00000000      2,00000000      0,00000000
3,0       3,00000000      3,00000000      0,00000000
4,0       4,00000000      4,00000000      0,00000000
5,0       5,00000000      5,00000000      0,00000000
6,0       6,00000000      6,00000000      0,00000000
7,0       7,00000000      7,00000000      0,00000000
8,0       8,00000000      8,00000000      0,00000000
9,0       9,00000000      9,00000000      0,00000000
10,0      10,00000000     10,00000000     0,00000000

Размер файла Externalizable: 236 байт
Временный файл удален


3. Сравнение методов сериализации
==========================================
Размер файла Serializable: 236 байт
Размер файла Externalizable: 236 байт
Разница: 0 байт

Анализ содержимого файлов:
Serializable файл содержит служебные данные Java
Externalizable файл содержит только данные точек

Временные файлы удалены
PS D:\Other\Study\GitHubLab4>
```