Лабораторная работа 4

Выполнил: студент группы

6301-030301D

Дымченко В. Р.

Задание 1

В классах ArrayTabulatedFunction и LinkedListTabulatedFunction добавим 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.

```
    public ArrayTabulatedFunction(FunctionPoint[] points) {  5 usages
    if (points.length < 2) {
        throw new IllegalArgumentException("Нужно не меньше 2 точек");
    }

    for (int i = 1; i < points.length; i++) {
        if (points[i].getX() - points[i-1].getX() <= -EPSILON) {
            throw new IllegalArgumentException("Точки должны быть упорядочены по координате x");
        }
    }

    this.pointsCount = points.length;
    this.points = new FunctionPoint[points.length + 5];
    for (int i = 0; i < points.length; i++) {
        this.points[i] = new FunctionPoint(points[i]);
    }
}
```

```
public LinkedListTabulatedFunction(FunctionPoint[] points) {  2 usages
    if (points.length < 2) {
        throw new IllegalArgumentException("Нужно не меньше 2 точек");
    }

    for (int i = 1; i < points.length; i++) {
        if (points[i].getX() - points[i-1].getX() <= -EPSILON) {
            throw new IllegalArgumentException("Точки должны быть упорядочены по координате x");
        }
    }

    initializeList();
    for (FunctionPoint point : points) {
        addNodeToTail().point = new FunctionPoint(point);
    }
}
```

Задание 2

В пакете functions создаем интерфейс Function, описывающий функции одной переменной и содержащий следующие методы:

• public double getLeftDomainBorder() – возвращает значение левой границы области определения функции;

• public double getRightDomainBorder() – возвращает значение правой границы области определения функции;

• public double getFunctionValue(double x) – возвращает значение функции в заданной точке.

Интерфейс TabulatedFunction будет теперь наследоваться от интерфейса Function.

```
package functions;

public interface Function {  15 implementations
    double getLeftDomainBorder();  11 implementations
    double getRightDomainBorder();  11 implementations
    double getFunctionValue(double x);  14 implementations
}
```

Задание 3-4

Создаем пакет functions.basic, в нём описываем классы ряда функций, заданных аналитически. Создаем класс TrigonometricFunction, реализующий интерфейс Function и описывающий методы получения границ области определения. Создаем наследующие от него публичные классы Sin, Cos и Tan .

Также создаем пакет functions.meta, в нём описываем классы функций, позволяющие комбинировать функции. Классы наследуются от интерфейса Function.

Задание 5

В пакете functions создаем класс Functions, содержащий вспомогательные статические методы для работы с функциями. Делаем так, чтобы в программе вне этого класса нельзя было создать его объект. При написании методов пользуемся созданными ранее классами из пакета functions.meta.

```java
package functions;

import functions.meta.*;

public final class Functions {  10 usages
    private Functions() {  no usages
        throw new UnsupportedOperationException("Cannot instantiate utility class");
    }

    public static Function shift(Function f, double shiftX, double shiftY) {  no usages
        return new Shift(f, shiftX, shiftY);
    }

    public static Function scale(Function f, double scaleX, double scaleY) {  1 usage
        return new Scale(f, scaleX, scaleY);
    }

    public static Function power(Function f, double power) {  4 usages
        return new Power(f, power);
    }

    public static Function sum(Function f1, Function f2) {  3 usages
        return new Sum(f1, f2);
    }

    public static Function mult(Function f1, Function f2) {  no usages
        return new Mult(f1, f2);
    }

    public static Function composition(Function f1, Function f2) {  2 usages
        return new Composition(f1, f2);
    }
```

Задание 6

В пакете functions создаем класс TabulatedFunctions, содержащий вспомогательные статические методы для работы с табулированными функциями. Описываем в классе 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. Если указанные границы для табулирования выходят за область определения функции, метод должен выбрасывать исключение IllegalArgumentException.

```
public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
    if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
        throw new IllegalArgumentException("Границы табуляции выходят за область определения функции");
    }
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Требуется не менее 2 точек");
    }

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

    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + i * step;
        double y = function.getFunctionValue(x);
        points[i] = new FunctionPoint(x, y);
    }

    return new ArrayTabulatedFunction(points);
}
```

Задание 7

В класс TabulatedFunctions добавьте следующие методы.

Метод вывода табулированной функции в байтовый поток public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out).

Метод ввода табулированной функции из байтового потока public static TabulatedFunction inputTabulatedFunction(InputStream in).

Метод записи табулированной функции в символьный поток public static void writeTabulatedFunction(TabulatedFunction function, Writer out).

Метод чтения табулированной функции из символьного потока public static TabulatedFunction readTabulatedFunction(Reader in).

При написании методов в первых трёх случаях необходимо воспользоваться DataOutputStream, DataInputStream, PrintWriter, а в четвёртом случае – классом StreamTokenizer.

```java
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) {  1 usage
    try (DataOutputStream dos = new DataOutputStream(out)) {
        dos.writeInt(function.getPointsCount());
        for (int i = 0; i < function.getPointsCount(); i++) {
            FunctionPoint point = function.getPoint(i);
            dos.writeDouble(point.getX());
            dos.writeDouble(point.getY());
        }
        dos.flush();
    } catch (IOException e) {
        throw new RuntimeException("Ошибка записи табулированной функции в поток", e);
    }
}

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

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

        return new ArrayTabulatedFunction(points);
    } catch (IOException e) {
        throw new RuntimeException("Ошибка чтения табулированной функции из потока", e);
    }
```

```java
public static TabulatedFunction readTabulatedFunction(Reader in) {  1 usage
    try {
        StreamTokenizer tokenizer = new StreamTokenizer(in);
        tokenizer.parseNumbers();


        if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
            throw new RuntimeException("Ожидалось количество точек");
        }
        int pointsCount = (int) tokenizer.nval;

        FunctionPoint[] points = new FunctionPoint[pointsCount];


        for (int i = 0; i < pointsCount; i++) {
            if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
                throw new RuntimeException("Ожидалась координата X");
            }
            double x = tokenizer.nval;

            if (tokenizer.nextToken() != StreamTokenizer.TT_NUMBER) {
                throw new RuntimeException("Ожидалась координата Y");
            }
            double y = tokenizer.nval;

            points[i] = new FunctionPoint(x, y);
        }
```

Задание 8

Проверим все написанные классы и методы:

```
Синус и косинус на [0, π]:
sin(0,0) = 0,0000, cos(0,0) = 1,0000
sin(0,1) = 0,0998, cos(0,1) = 0,9950
sin(0,2) = 0,1987, cos(0,2) = 0,9801
sin(0,3) = 0,2955, cos(0,3) = 0,9553
sin(0,4) = 0,3894, cos(0,4) = 0,9211
sin(0,5) = 0,4794, cos(0,5) = 0,8776
sin(0,6) = 0,5646, cos(0,6) = 0,8253
sin(0,7) = 0,6442, cos(0,7) = 0,7648
sin(0,8) = 0,7174, cos(0,8) = 0,6967
sin(0,9) = 0,7833, cos(0,9) = 0,6216
sin(1,0) = 0,8415, cos(1,0) = 0,5403
sin(1,1) = 0,8912, cos(1,1) = 0,4536
sin(1,2) = 0,9320, cos(1,2) = 0,3624
sin(1,3) = 0,9636, cos(1,3) = 0,2675
sin(1,4) = 0,9854, cos(1,4) = 0,1700
sin(1,5) = 0,9975, cos(1,5) = 0,0707
sin(1,6) = 0,9996, cos(1,6) = -0,0292
sin(1,7) = 0,9917, cos(1,7) = -0,1288
sin(1,8) = 0,9738, cos(1,8) = -0,2272
sin(1,9) = 0,9463, cos(1,9) = -0,3233
sin(2,0) = 0,9093, cos(2,0) = -0,4161
sin(2,1) = 0,8632, cos(2,1) = -0,5048
sin(2,2) = 0,8085, cos(2,2) = -0,5885
sin(2,3) = 0,7457, cos(2,3) = -0,6663
sin(2,4) = 0,6755, cos(2,4) = -0,7374
sin(2,5) = 0,5985, cos(2,5) = -0,8011
sin(2,6) = 0,5155, cos(2,6) = -0,8569
sin(2,7) = 0,4274, cos(2,7) = -0,9041
sin(2,8) = 0,3350, cos(2,8) = -0,9422
sin(2,9) = 0,2392, cos(2,9) = -0,9710
```

```
Сумма квадратов табулированных синуса и косинуса:
x=0,0: sin²+cos²=1,0000 (теоретически=1.0000)
x=0,1: sin²+cos²=0,9753 (теоретически=1.0000)
x=0,2: sin²+cos²=0,9705 (теоретически=1.0000)
x=0,3: sin²+cos²=0,9854 (теоретически=1.0000)
x=0,4: sin²+cos²=0,9850 (теоретически=1.0000)
x=0,5: sin²+cos²=0,9704 (теоретически=1.0000)
x=0,6: sin²+cos²=0,9756 (теоретически=1.0000)
x=0,7: sin²+cos²=0,9994 (теоретически=1.0000)
x=0,8: sin²+cos²=0,9751 (теоретически=1.0000)
x=0,9: sin²+cos²=0,9706 (теоретически=1.0000)
x=1,0: sin²+cos²=0,9859 (теоретически=1.0000)
x=1,1: sin²+cos²=0,9845 (теоретически=1.0000)
x=1,2: sin²+cos²=0,9703 (теоретически=1.0000)
x=1,3: sin²+cos²=0,9759 (теоретически=1.0000)
x=1,4: sin²+cos²=0,9987 (теоретически=1.0000)
x=1,5: sin²+cos²=0,9748 (теоретически=1.0000)
x=1,6: sin²+cos²=0,9707 (теоретически=1.0000)
x=1,7: sin²+cos²=0,9864 (теоретически=1.0000)
x=1,8: sin²+cos²=0,9841 (теоретически=1.0000)
x=1,9: sin²+cos²=0,9702 (теоретически=1.0000)
x=2,0: sin²+cos²=0,9762 (теоретически=1.0000)
x=2,1: sin²+cos²=0,9981 (теоретически=1.0000)
x=2,2: sin²+cos²=0,9745 (теоретически=1.0000)
x=2,3: sin²+cos²=0,9708 (теоретически=1.0000)
x=2,4: sin²+cos²=0,9869 (теоретически=1.0000)
x=2,5: sin²+cos²=0,9836 (теоретически=1.0000)
x=2,6: sin²+cos²=0,9702 (теоретически=1.0000)
x=2,7: sin²+cos²=0,9765 (теоретически=1.0000)
x=2,8: sin²+cos²=0,9975 (теоретически=1.0000)
```

```
Тестирование ввода-вывода:
Байтовые потоки (экспонента):
Сравнение исходной и прочитанной экспоненты:
x=0,0: исходная=1,0000, прочитанная=1,0000, true
x=1,0: исходная=2,7183, прочитанная=2,7183, true
x=2,0: исходная=7,3891, прочитанная=7,3891, true
x=3,0: исходная=20,0855, прочитанная=20,0855, true
x=4,0: исходная=54,5982, прочитанная=54,5982, true
x=5,0: исходная=148,4132, прочитанная=148,4132, true
x=6,0: исходная=403,4288, прочитанная=403,4288, true
x=7,0: исходная=1096,6332, прочитанная=1096,6332, true
x=8,0: исходная=2980,9580, прочитанная=2980,9580, true
x=9,0: исходная=8103,0839, прочитанная=8103,0839, true
x=10,0: исходная=22026,4658, прочитанная=22026,4658, true
Все значения совпадают: true

Размер бинарного файла: 180 байт

Символьные потоки (логарифм):
Сравнение исходного и прочитанного логарифма:
x=0,1: исходный=-2,3026, прочитанный=-2,3026, true
x=1,1: исходный=0,0927, прочитанный=0,0927, true
x=2,1: исходный=0,7402, прочитанный=0,7402, true
x=3,1: исходный=1,1301, прочитанный=1,1301, true
x=4,1: исходный=1,4100, прочитанный=1,4100, true
x=5,1: исходный=1,6284, прочитанный=1,6284, true
x=6,1: исходный=1,8076, прочитанный=1,8076, true
x=7,1: исходный=1,9595, прочитанный=1,9595, true
x=8,1: исходный=2,0913, прочитанный=2,0913, true
x=9,1: исходный=2,2078, прочитанный=2,2078, true
Все значения совпадают: true
```

```
Тестирование meta функций:
Композиция sin(cos(x)):
x=0,0: sin(cos(x))=0,8415
x=0,5: sin(cos(x))=0,7692
x=1,0: sin(cos(x))=0,5144
x=1,5: sin(cos(x))=0,0707
x=2,0: sin(cos(x))=-0,4042
x=2,5: sin(cos(x))=-0,7182
x=3,0: sin(cos(x))=-0,8360

Сумма sin(x) + cos(x):
x=0,0: sin+cos=1,0000
x=0,5: sin+cos=1,3570
x=1,0: sin+cos=1,3818
x=1,5: sin+cos=1,0682
x=2,0: sin+cos=0,4932
x=2,5: sin+cos=-0,2027
x=3,0: sin+cos=-0,8489

Масштабированный синус 3*sin(2x):
x=0,0: 3*sin(2x)=0,0000
x=0,5: 3*sin(2x)=2,5244
x=1,0: 3*sin(2x)=2,7279
x=1,5: 3*sin(2x)=0,4234
x=2,0: 3*sin(2x)=-2,2704
x=2,5: 3*sin(2x)=-2,8768
x=3,0: 3*sin(2x)=-0,8382
```

Задание 9

Сделайте так, чтобы объекты всех классов, реализующих интерфейс TabulatedFunction, были сериализуемыми.

Для этого рассмотрите два случая:

1. с использованием интерфейса java.io.Serializable

   Тут просто добавляем к сериализуемым классам интерфейс Serializable

   ```
   public class ArrayTabulatedFunction implements TabulatedFunction, Serializable {  10 usages
   ```

2. с использованием интерфейса java.io.Externalizable

В этом случае добавим к сериализуемым классам интерфейс Externalizable, конструкторы и спец. методы для ArrayTabulatedFunction, LinkedListTabulatedFunction и FunctionPoint .

```
public void writeExternal(ObjectOutput out) throws IOException {  no
    out.writeInt(pointsCount);
    for (int i = 0; i < pointsCount; i++) {
        out.writeDouble(points[i].getX());
        out.writeDouble(points[i].getY());
    }
}

public void readExternal(ObjectInput in) throws IOException, ClassNo
    pointsCount = in.readInt();
    points = new FunctionPoint[pointsCount + 5];
    for (int i = 0; i < pointsCount; i++) {
        double x = in.readDouble();
        double y = in.readDouble();
        points[i] = new FunctionPoint(x, y);
    }
}
```

```java
public void writeExternal(ObjectOutput out) throws IOException {  no usag
    out.writeInt(pointsCount);

    FunctionNode node = head.next;
    while (node != head) {
        out.writeDouble(node.point.getX());
        out.writeDouble(node.point.getY());
        node = node.next;
    }
}

public void readExternal(ObjectInput in) throws IOException, ClassNotFo
    int count = in.readInt();
    initializeList();

    for (int i = 0; i < count; i++) {
        double x = in.readDouble();
        double y = in.readDouble();
        addNodeToTail().point = new FunctionPoint(x, y);
    }
}
```

```
Тестирование сериализации

Тестирование Serializable:
Исходная функция (ln(exp(x))):
  (0,0, 0,0)  (1,0, 1,0)  (2,0, 2,0)  (3,0, 3,0)  (4,0, 4,0)  (5,0, 5,0)  (6,0, 6,0)  (7,0, 7,0)  (8,0, 8,0)  (9,0, 9,0)  (10,0, 10,0)
Сериализация
Десериализация
Сравнение значений:
  x=0,0: 0,0 и 0,0 - Совпало
  x=1,0: 1,0 и 1,0 - Совпало
  x=2,0: 2,0 и 2,0 - Совпало
  x=3,0: 3,0 и 3,0 - Совпало
  x=4,0: 4,0 и 4,0 - Совпало
  x=5,0: 5,0 и 5,0 - Совпало
  x=6,0: 6,0 и 6,0 - Совпало
  x=7,0: 7,0 и 7,0 - Совпало
  x=8,0: 8,0 и 8,0 - Совпало
  x=9,0: 9,0 и 9,0 - Совпало
  x=10,0: 10,0 и 10,0 - Совпало
Результат: Правильно
Размер файла: 445 байт
```

```
Тестирование Externalizable:
Исходная функция sin(x):
  (0,00, 0,0000)  (0,35, 0,3420)  (0,70, 0,6428)  (1,05, 0,8660)  (1,40, 0,9848)  (1,75, 0,9848)  (2,09, 0,8660)  (2,44, 0,6428)  (2,79, 0,3420)  (3,14, 0,0000)
Сериализация с Externalizable
Десериализация
Сравнение значений:
  x=0,0: 0,0000 и 0,0000 - Совпало
  x=0,5: 0,4721 и 0,4721 - Совпало
  x=1,0: 0,8358 и 0,8358 - Совпало
  x=1,5: 0,9848 и 0,9848 - Совпало
  x=2,0: 0,8981 и 0,8981 - Совпало
  x=2,5: 0,5941 и 0,5941 - Совпало
  x=3,0: 0,1387 и 0,1387 - Совпало
Результат: Правильно
Размер файла: 423 байт

Сравнение файлов:
Serializable: 445 байт
Externalizable: 423 байт
Разница: 22 байт
Externalizable создает меньшие файлы
```

Изменения TestSerialization

Создаем одну и ту же функцию - табулированный аналог логарифма по натуральному основанию, взятого от экспоненты на отрезке от 0 до 10 с 11 точками. Явно создаем ArrayTabulatedFunction - он будет использовать Serializable, и LinkedListTabulatedFunction - он будет использовать Externalizable.

```java
private static void testSerializableVsExternalizable() {  1 usage   Vlados-ux *
    System.out.println("Создание и сериализация ln(exp(x)) на [0, 10] с 11 точками:");

    try {
        Exp exp = new Exp();
        Log log = new Log(Math.E);
        Function composition = Functions.composition(exp, log);

        FunctionPoint[] points = new FunctionPoint[11];
        double step = 10.0 / 10;

        System.out.println("Создаем точки функции ln(exp(x)) = x:");
        for (int i = 0; i < 11; i++) {
            double x = i * step;  // x = 0, 1, 2, ..., 10
            double y = composition.getFunctionValue(x);  // y = ln(exp(x)) = x
            points[i] = new FunctionPoint(x, y);
            System.out.printf("  Точка %2d: (%.1f, %.1f)%n", i, x, y);
        }
```

```java
System.out.println("\nСоздание ArrayTabulatedFunction:");
ArrayTabulatedFunction arrayFunc = new ArrayTabulatedFunction(points);
System.out.println("Array функция создана, точек: " + arrayFunc.getPointsCount());

System.out.println("Создание LinkedListTabulatedFunction:");
LinkedListTabulatedFunction linkedListFunc = new LinkedListTabulatedFunction(points);
System.out.println("LinkedList функция создана, точек: " + linkedListFunc.getPointsCount());

System.out.println("\nТест ArrayTabulatedFunction (Serializable)");
testSerializationMethod(arrayFunc, filename: "array_serializable.dat", methodName: "Array Serializable");

System.out.println("\nТест LinkedListTabulatedFunction (Externalizable)");
testSerializationMethod(linkedListFunc, filename: "linkedlist_externalizable.dat", methodName: "LinkedList Externalizable");

compareFiles();
```

Тестирование

```
Тестирование сериализации

Создание и сериализация ln(exp(x)) на [0, 10] с 11 точками:
Создаем точки функции ln(exp(x)) = x:
  Точка  0: (0,0, 0,0)
  Точка  1: (1,0, 1,0)
  Точка  2: (2,0, 2,0)
  Точка  3: (3,0, 3,0)
  Точка  4: (4,0, 4,0)
  Точка  5: (5,0, 5,0)
  Точка  6: (6,0, 6,0)
  Точка  7: (7,0, 7,0)
  Точка  8: (8,0, 8,0)
  Точка  9: (9,0, 9,0)
  Точка 10: (10,0, 10,0)

Создание ArrayTabulatedFunction:
Array функция создана, точек: 11
Создание LinkedListTabulatedFunction:
LinkedList функция создана, точек: 11

Тест: ArrayTabulatedFunction
ArrayTabulatedFunction - Сериализация в файл: array_serializable.dat
ArrayTabulatedFunction - Десериализация из файла: array_serializable.dat
ArrayTabulatedFunction - Проверка корректности данных
ArrayTabulatedFunction - Результат: Успешно
ArrayTabulatedFunction - Размер файла: 445 байт
```

```
Тест: LinkedListTabulatedFunction
LinkedListTabulatedFunction - Сериализация в файл: linkedlist_externalizable.dat
LinkedListTabulatedFunction - Десериализация из файла: linkedlist_externalizable.dat
LinkedListTabulatedFunction - Проверка корректности данных
LinkedListTabulatedFunction - Результат: Успешно
LinkedListTabulatedFunction - Размер файла: 241 байт

Сравнение размеров файлов
ArrayTabulatedFunction: 445 байт
LinkedListTabulatedFunction: 241 байт
Разница: 204 байт
Вывод: LinkedList (Externalizable) эффективнее на 204 байт

Тестирование завершено

Process finished with exit code 0
```