Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №4

Дисциплина: «ООП»

Тема «Расширить возможности пакета для работы с функциями одной переменной добавив интерфейсы и классы для аналитически заданных функций, а также методы ввода и вывода табулированных функций.»

Выполнил: Зорин Дмитрий Сергеевич

Группа: 6201-120303D

Самара, 2025

Задание на лабораторную работу

Задание 1

В классах ArrayTabulatedFunction и LinkedListTabulatedFunction я добавил конструктор, который принимает массив точек FunctionPoint[]. Если точек меньше двух или точки не упорядочены, то выбрасывается исключение IllegalArgumentException. Пример кода:

```
public LinkedListTabulatedFunction(FunctionPoint[] points) {  2 usages
    if (points == null) {
        throw new IllegalArgumentException("Массив null");
    }
    if (points.length < 2) {
        throw new IllegalArgumentException("Мало точек");
    }

    for (int i = 0; i < points.length - 1; i++) {
        if (points[i].getX() >= points[i + 1].getX()) {
            throw new IllegalArgumentException("Неупорядочены по x");
        }
    }
```

Задание 2

Во 2 задании я создал интерфейс Function для описания функции 1 переменной с различными методами:

```
package functions;

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

Теперь TabulatedFunction расширяет Function

Задание 3

В пакете functions.basic я описал классы ряда функции

Экспонента:

```java
package functions.basic;

import functions.Function;

public class Exp implements Function {  1 usage

    @Override
    public double getLeftDomainBorder() {
        return -Double.MAX_VALUE; // граница области определения экспоненты
    }

    @Override
    public double getRightDomainBorder() {
        return Double.MAX_VALUE;
    }

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

Логарифм по основанию:

```java
package functions.basic;

import functions.Function;

public class Log implements Function {  1 usage
    private double base;  2 usages

    public Log(double base) {  3 usages
        if (base <= 0 || base == 1) {
            throw new IllegalArgumentException("Основание логарифма должно быть >0 и !=1");
        }
        this.base = base;
    }

    @Override
    public double getLeftDomainBorder() {
        return 0; // логарифм определён только для положительных x
    }

    @Override
    public double getRightDomainBorder() {
        return Double.MAX_VALUE;
    }

    @Override
    public double getFunctionValue(double x) {
        if (x <= 0) {
            return Double.NaN; // логарифм отрицательного числа не определён
        }
        return Math.log(x) / Math.log(base);
    }
}
```

Тригонометрические функции:

```java
package functions.basic;

import functions.Function;

public abstract class TrigonometricFunction implements Function {  3 usages

    @Override
    public double getLeftDomainBorder() {
        return -Double.MAX_VALUE;
    }

    @Override
    public double getRightDomainBorder() {
        return Double.MAX_VALUE;
    }
}
```

Далее я создал наследующие классы sin,cos,tan Пример

кода для sin:

```java
package functions.basic;

public class Sin extends TrigonometricFunction {  no usages

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

Задание 4

Мы создали пакет functions.meta с классами, которые позволяют строить новые функции на основе уже существующих:

Sum - сумма двух функций, область определения

Mult - произведение двух функций, область определения

Power - возведение функции в степень, область определения совпадает с базовой функцией

Scale - масштабирование функции по осям X и Y, область и значения корректно масштабируются

Shift - сдвиг функции по осям X и Y, область и значения смещаются на заданные величины
Composition - композиция функций (f(g(x))), область определяется внутренней функцией

Примеры кода различных классов:

Sum:

```java
package functions.meta;

import functions.Function;

public class Sum implements Function {  1 usage

    private Function f1;  4 usages
    private Function f2;  4 usages

    public Sum(Function f1, Function f2) {  3 usages
        this.f1 = f1;
        this.f2 = f2;
    }

    @Override
    public double getLeftDomainBorder() {
        return Math.max(f1.getLeftDomainBorder(), f2.getLeftDomainBorder());
    }

    @Override
    public double getRightDomainBorder() {
        return Math.min(f1.getRightDomainBorder(), f2.getRightDomainBorder());
    }

    @Override
    public double getFunctionValue(double x) {
        return f1.getFunctionValue(x) + f2.getFunctionValue(x);
    }
}
```

Power:

```java
package functions.meta;

import functions.Function;

public class Power implements Function {  1 usage

    private Function f;  4 usages
    private double power;  2 usages

    public Power(Function f, double power) {  3 usages
        this.f = f;
        this.power = power;
    }

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

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

    @Override
    public double getFunctionValue(double x) {
        return Math.pow(f.getFunctionValue(x), power);
    }
}
```

Все эти классы реализуют интерфейс Function

Задание 5

Мы создали класс Functions, который предоставляет статические методы для работы с функциями и позволяет создавать новые функции на основе уже существующих
Пример кода:

```java
package functions;

import functions.meta.*;

public final class Functions {  no usages
    private Functions() { }  no usages

    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) {  no usages
        return new Scale(f, scaleX, scaleY);
    }

    public static Function power(Function f, double power) {  no usages
        return new Power(f, power);
    }

    public static Function sum(Function f1, Function f2) {  no 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) {  no usages
        return new Composition(f1, f2);
    }
}
```

В классе необходимо было описать методы, например:

public static Function shift(Function f, double shiftX, double shiftY) – возвращает объект
функции, полученной из исходной сдвигом вдоль осей

public static Function sum(Function f1, Function f2) – возвращает объект функции, являющейся суммой двух исходных

Задание 6

В пакете functions создан класс TabulatedFunctions, предназначенный для работы с табулированными функциями. Класс содержит только статические методы и не допускает создание экземпляров вне этого класса Основной метод класса:

```java
public static TabulatedFunction tabulate(Function f, double leftX, double rightX, int pointsCount)
```

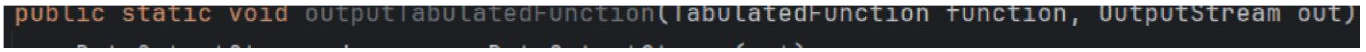
Он принимает объект Function и возвращает табулированный аналог этой функции,

Если границы выходят за область, то выбрасывается исключение

Задание 7

В класс TabulatedFunctions добавлены методы для сериализации/десериализации табулированных функций в байтовые и символьные потоки.

Вывод табулированной функции в байтовый поток:



Метод записывает в поток количество точек функции и координаты каждой точки (x и y).

Ввод табулированной функции из байтового потока

Метод считывает данные о функции

Обработка IOException

Все методы объявляют throws IOException. Это позволяет передавать ответственность за обработку исключений вызывающему коду

Закрытие потоков

Потоки, переданные в методы извне, не закрываются внутри методов, т.к это ответственность вызывающего кода

Задание 8

Работа с Sin и Cos
Сначала создаем объекты

```
Sin sinFunc = new Sin();
Cos cosFunc = new Cos();
```

Далее выводим значения на отрезке [0, Pi] с шагом 0.1:

```
for (double x = 0; x <= Math.PI; x += 0.1) {
    System.out.printf("x=%.1f: sin=%.4f, cos=%.4f%n",
            x, sinFunc.getFunctionValue(x), cosFunc.getFunctionValue(x));
}
```

Табулируем с помощью TabulatedFunctions.tabulate():

```
TabulatedFunction tabSin = TabulatedFunctions.tabulate(sinFunc, leftX: 0, Math.PI, pointsCount: 10);
TabulatedFunction tabCos = TabulatedFunctions.tabulate(cosFunc, leftX: 0, Math.PI, pointsCount: 10);
```

```
Function sumSquares = Functions.sum(
        Functions.power(tabSin, power: 2),
        Functions.power(tabCos, power: 2)
);
System.out.println("\nСумма квадратов табулированных функций:");
for (double x = 0; x <= Math.PI; x += 0.1) {
    System.out.printf("x=%.1f: sumSquares=%.4f%n", x, sumSquares.getFunctionValue(x));
}
```

Создаём функцию суммы квадратов через Functions и выводим на [0, Pi] с шагом 0.1

Далее с помощью метода TabulatedFunctions.tabulate() создаем табулированный аналог экспоненты и табулированный аналог логарифма

Задание 9

Нам необходимо сделать все классы, реализующие TabulatedFunction, сериализуемыми

Создаем табулированный логарифм от экспоненты на отрезке [0, 10] с 11

точками:

```
Exp expFunc = new Exp();
Log lnFunc = new Log(Math.E);
Function lnOfExpFunc = new Composition(lnFunc, expFunc);

TabulatedFunction lnOfExp = TabulatedFunctions.tabulate(lnOfExpFunc, 0, 10, 11);
```

Сериализация в файл lnOfExp_ser.ser:

```
try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("lnOfExp.ser"))) {
   oos.writeObject(lnOfExp);
}
```

Десериализация из файла:

```
TabulatedFunction readLnOfExp;
try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("lnOfExp.ser"))) {
   readLnOfExp = (TabulatedFunction) ois.readObject();
}
```

Пример работы кода:

```
Поймано IllegalArgumentException: Левая граница должна быть < правой
Поймано IllegalArgumentException: Количество точек должно быть ≥ 2
Ошибка при добавлении новой точки: Точка с таким X уже существует
Y второй точки изменено на 10
Значение функции при x = 2.5: 7.0

Текущее состояние функции:
```

(2.0, 4.0)
(3.0, 10.0)
(4.0, 16.0)

Sin и Cos на [0, π] с шагом 0.1:
x=0,0: sin=0,0000, cos=1,0000
x=0,1: sin=0,0998, cos=0,9950
x=0,2: sin=0,1987, cos=0,9801
x=0,3: sin=0,2955, cos=0,9553
x=0,4: sin=0,3894, cos=0,9211
x=0,5: sin=0,4794, cos=0,8776
x=0,6: sin=0,5646, cos=0,8253
x=0,7: sin=0,6442, cos=0,7648
x=0,8: sin=0,7174, cos=0,6967
x=0,9: sin=0,7833, cos=0,6216
x=1,0: sin=0,8415, cos=0,5403
x=1,1: sin=0,8912, cos=0,4536
x=1,2: sin=0,9320, cos=0,3624
x=1,3: sin=0,9636, cos=0,2675
x=1,4: sin=0,9854, cos=0,1700
x=1,5: sin=0,9975, cos=0,0707
x=1,6: sin=0,9996, cos=-0,0292
x=1,7: sin=0,9917, cos=-0,1288
x=1,8: sin=0,9738, cos=-0,2272
x=1,9: sin=0,9463, cos=-0,3233
x=2,0: sin=0,9093, cos=-0,4161
x=2,1: sin=0,8632, cos=-0,5048
x=2,2: sin=0,8085, cos=-0,5885
x=2,3: sin=0,7457, cos=-0,6663
x=2,4: sin=0,6755, cos=-0,7374
x=2,5: sin=0,5985, cos=-0,8011
x=2,6: sin=0,5155, cos=-0,8569
x=2,7: sin=0,4274, cos=-0,9041
x=2,8: sin=0,3350, cos=-0,9422
x=2,9: sin=0,2392, cos=-0,9710
x=3,0: sin=0,1411, cos=-0,9900
x=3,1: sin=0,0416, cos=-0,9991

Табулированные значения Sin и Cos:

x=0,0: sin=0,0000, cos=1,0000
x=0,1: sin=0,0980, cos=0,9827
x=0,2: sin=0,1960, cos=0,9654
x=0,3: sin=0,2939, cos=0,9482
x=0,4: sin=0,3859, cos=0,9144
x=0,5: sin=0,4721, cos=0,8646
x=0,6: sin=0,5582, cos=0,8149
x=0,7: sin=0,6440, cos=0,7646
x=0,8: sin=0,7079, cos=0,6884
x=0,9: sin=0,7719, cos=0,6122
x=1,0: sin=0,8358, cos=0,5360
x=1,1: sin=0,8840, cos=0,4506
x=1,2: sin=0,9180, cos=0,3571
x=1,3: sin=0,9521, cos=0,2636
x=1,4: sin=0,9848, cos=0,1699
x=1,5: sin=0,9848, cos=0,0704
x=1,6: sin=0,9848, cos=-0,0291
x=1,7: sin=0,9848, cos=-0,1285
x=1,8: sin=0,9662, cos=-0,2248
x=1,9: sin=0,9322, cos=-0,3183
x=2,0: sin=0,8981, cos=-0,4117
x=2,1: sin=0,8624, cos=-0,5043
x=2,2: sin=0,7985, cos=-0,5805
x=2,3: sin=0,7345, cos=-0,6567
x=2,4: sin=0,6706, cos=-0,7329
x=2,5: sin=0,5941, cos=-0,7942
x=2,6: sin=0,5079, cos=-0,8439
x=2,7: sin=0,4217, cos=-0,8937
x=2,8: sin=0,3347, cos=-0,9410
x=2,9: sin=0,2367, cos=-0,9583
x=3,0: sin=0,1387, cos=-0,9755
x=3,1: sin=0,0408, cos=-0,9928

Сумма квадратов табулированных функций:
x=0,0: sumSquares=1,0000
x=0,1: sumSquares=0,9753
x=0,2: sumSquares=0,9705
x=0,3: sumSquares=0,9854
x=0,4: sumSquares=0,9850

x=0,5: sumSquares=0,9704
x=0,6: sumSquares=0,9756
x=0,7: sumSquares=0,9994
x=0,8: sumSquares=0,9751
x=0,9: sumSquares=0,9706
x=1,0: sumSquares=0,9859
x=1,1: sumSquares=0,9845
x=1,2: sumSquares=0,9703
x=1,3: sumSquares=0,9759
x=1,4: sumSquares=0,9987
x=1,5: sumSquares=0,9748
x=1,6: sumSquares=0,9707
x=1,7: sumSquares=0,9864
x=1,8: sumSquares=0,9841
x=1,9: sumSquares=0,9702
x=2,0: sumSquares=0,9762
x=2,1: sumSquares=0,9981
x=2,2: sumSquares=0,9745
x=2,3: sumSquares=0,9708
x=2,4: sumSquares=0,9869
x=2,5: sumSquares=0,9836
x=2,6: sumSquares=0,9702
x=2,7: sumSquares=0,9765
x=2,8: sumSquares=0,9975
x=2,9: sumSquares=0,9743
x=3,0: sumSquares=0,9709
x=3,1: sumSquares=0,9873

Сравнение исходной и считанной экспоненты:
x=0: original=1,0000, read=1,0000
x=1: original=2,7183, read=2,7183
x=2: original=7,3891, read=7,3891
x=3: original=20,0855, read=20,0855
x=4: original=54,5982, read=54,5982
x=5: original=148,4132, read=148,4132
x=6: original=403,4288, read=403,4288
x=7: original=1096,6332, read=1096,6332
x=8: original=2980,9580, read=2980,9580
x=9: original=8103,0839, read=8103,0839

x=10: original=22026,4658, read=22026,4658

Сравнение исходного и считанного логарифма (бинарный файл):
x=0: original=NaN, read=NaN
x=1: original=0,0000, read=0,0000
x=2: original=0,6849, read=0,6849
x=3: original=1,0916, read=1,0916
x=4: original=1,3809, read=1,3809
x=5: original=1,6055, read=1,6055
x=6: original=1,7889, read=1,7889
x=7: original=1,9440, read=1,9440
x=8: original=2,0783, read=2,0783
x=9: original=2,1967, read=2,1967
x=10: original=2,3026, read=2,3026

Сравнение композиции ln(exp(x)):
x=0: ln(exp(x))=0,0000
x=1: ln(exp(x))=1,0000
x=2: ln(exp(x))=2,0000
x=3: ln(exp(x))=3,0000
x=4: ln(exp(x))=4,0000
x=5: ln(exp(x))=5,0000
x=6: ln(exp(x))=6,0000
x=7: ln(exp(x))=7,0000
x=8: ln(exp(x))=8,0000
x=9: ln(exp(x))=9,0000
x=10: ln(exp(x))=10,0000

Сравнение исходного и считанного ln(exp(x)) после сериализации:
x=0: original=0,0000, read=0,0000
x=1: original=1,0000, read=1,0000
x=2: original=2,0000, read=2,0000
x=3: original=3,0000, read=3,0000
x=4: original=4,0000, read=4,0000
x=5: original=5,0000, read=5,0000
x=6: original=6,0000, read=6,0000
x=7: original=7,0000, read=7,0000
x=8: original=8,0000, read=8,0000
x=9: original=9,0000, read=9,0000

x=10: original=10,0000, read=10,0000