Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
Dr. J. Johannsen, Dr. R. Matthes

SS 2004

8.6.2004

Übungen zur Vorlesung
Informatik II

Blatt 7

Abgabe der Hausaufgaben bis spätestens am 21.6.2004, 14:00 Uhr über

`http://lehre.tcs.ifi.lmu.de/info2/Abgabe/abgabe.php`,

Bearbeitung in Gruppen zu max. 3 Personen ist zulässig.

Bitte halten Sie sich an folgende Grundsätze zur Bearbeitung der Aufgaben. Die Bewertung Ihrer Abgaben kann andernfalls deutlich herabgesetzt werden.

Verwenden Sie genau die Signatur der Aufgabenstellung, insbesondere die Namen der Klassen und Methoden in der vorgegebenen Klein- und Großschreibung.

Verwenden Sie keine „magischen" Größen in Ihren Programmen, d. h., weisen Sie Konstanten (außer den Zahlen 0 und 1 und dem leeren String) immer einer mit `final` deklarierten und in Großbuchstaben geschriebenen „Variablen" zu und verwenden Sie anschließend nur diese. Wer gute Gründe hat für eine Abweichung von dieser Regel, der kommentiere seinen Code entsprechend.

Jede Klasse und `public`-Methode muß angemessen mit `javadoc`-Kommentaren dokumentiert werden.

Programmieraufgabe P-16 (`LogApproximatorTest.java`, `Testausfuehrung.txt`, `LogApproximator.java`): **12 Punkte**

Auf der Vorlesungshomepage finden Sie den Quellcode für eine Klasse `LogApproximator`, die eine Näherung an den natürlichen Logarithmus eines `double`-Wertes berechnet. Dazu wird die Reihenentwicklung

$$\ln x = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(x-1)^n}{n}$$

verwendet. Da diese Reihe nur für $0 < x \leq 2$ konvergiert, und außerdem am Rand dieses Intervalls nur sehr langsam, werden alle Werte von x, die ausserhalb eines Intervalls $\delta < x < 2 - \delta$ für ein kleines $\delta > 0$ liegen, durch Multiplikation mit einer geeigneten Potenz von e skaliert. Der Logarithmus wird dann gemäß der Regel

$$\ln(x \cdot e^k) = \ln x + k$$

berechnet.

Schreiben Sie eine Testklasse `LogApproximatorTest` für diese Klasse unter Verwendung von JUnit. Erben Sie also von `junit.framework.TestCase`, überschreiben Sie `protected void setUp()` und `protected void tearDown()` und verwenden Sie die `main`-Methode

```
public static void main(String[] args){
    Class testklasse = LogApproximatorTest.class;
    junit.textui.TestRunner.run(testklasse);
}
```

Dies läßt sich nur kompilieren und ausführen, wenn `junit.jar` (z. B. von der Vorlesungshomepage geholt) im Klassenpfad liegt (kein Entpacken nötig!). Liegt diese Datei im Oberverzeichnis des aktuellen Verzeichnisses mit den Klassen der Aufgabe, so wird unter Unix die Testklasse mittels

```
javac -classpath .:../junit.jar LogApproximatorTest.java
```

kompiliert. Die Ausführung geht dann mittels

```
java -classpath .:../junit.jar LogApproximatorTest
```

Zu den Tests: Wir wollen 20 Eingabewerte testen. Davon sollen 3 fest vorgegeben werden (mindestens einer davon größer als 10^{23}) und die anderen einmal zufällig aus dem Bereich von 0 bis 2 gezogen werden. (Damit alle Tests mit denselben Zufallszahlen ausgeführt werden, muß der Zufallszahlengenerator der Klasse `java.util.Random` mit dem Konstruktor `Random(long startwert)` für einen festen Startwert erzeugt werden.) Implementieren Sie Methoden `public void testX()` mit passenden Namen anstelle von X. Jede soll genau eine der folgenden Eigenschaften testen:

1. Argument und Faktor werden im Konstruktor richtig aus dem Eingabewert berechnet, d. h., der Eingabewert kann aus Argument und Faktor wiedergewonnen werden.
2. Nach der Berechnung des Logarithmus liefert `nochUngenau()` den Wert `false`.
3. Wendet man `Math.exp` auf den berechneten Logarithmus an, so erhält man den Eingabewert zurück.
4. Die Methode `Math.ln` berechnet den gleichen Wert wie `getLog()`.

Benutzen Sie für den Vergleich von `double`-Werten eine sinnvolle Fehlerschranke. Da jede Ihrer Testmethoden alle 20 Eingabewerte durchgehen muß, soll im Fehlerfall auch ausgegeben werden, welcher Eingabewert gerade getestet wurde und der wievielte Eingabewert das war. Dazu bieten sich Methoden wie

```
public static void assertEquals(String, double, double, double)
```

aus der Klasse `junit.framework.Assert` an, deren Javadoc Teil von JUnit ist.

Testen Sie solange, bis Sie den Programmierfehler finden. Speichern Sie die Ausgabe der Testklasse mit den Fehlermeldungen in der Datei `Testausfuehrung.txt`. Beheben Sie den Fehler in `LogApproximator.java`. Überzeugen Sie sich, daß die Tests dann erfolgreich ablaufen. Abzugeben sind dann die Fehlermeldungen und die Sourcen der Testklasse und der korrigierten Klasse `LogApproximator`.

Programmieraufgabe P-17 (`geometrie2.jar`): **8 Punkte**
Erweitern Sie das Package `geometrie` aus Aufgabe P-15 um eine Klasse `Polygon`, deren Objekte beliebig lange geschlossene Polygonzüge repräsentieren. Dem Konstruktor der Klasse `Polygon` wird ein Array von Punkten übergeben, d.h. er soll wie folgt deklariert sein

```
public Polygon(Punkt[] punkte)
```

Implementieren Sie die Klasse `Polygon` als konkrete Unterklasse der Klasse `Figur`.

Ändern sie weiterhin in der alten Lösung die Klassen `Dreieck` und `Viereck` so ab, dass sie Unterklassen der Klasse `Polygon` werden. Lassen Sie die modifizierten Klassen `Dreieck` und `Viereck` so viele Methoden wie möglich von `Polygon` erben.

Betrachten Sie die Testbeispiele für Aufgabe P-15 mit den geänderten Klassen (keine Abgabe verlangt).

Speichern Sie das modifizierte Package (das weiterhin `geometrie` heißen soll) in einem JAR-Archiv `geometrie2.jar`, und geben Sie dieses ab. Falls Sie die Aufgabe P-15 nicht oder nicht korrekt bearbeitet haben, können Sie unseren Lösungsvorschlag als Grundlage benutzen, der ab dem 14.6. abends auf der Vorlesungshomepage verfügbar sein wird.