SoSe 2014
Prof. Dr. Margarita Esponda
ALP2
**6. Übungsblatt**

**Ziel**:
1) Auseinandersetzung mit dem Hoare-Kalkül und Terminierung von Schleifen.
2) Erste Auseinandersetzung mit der Java-Syntax und Semantik, grundlegende imperative Konzepte von Java, Java-Datentypen und einfache Pseudo-Klassendefinitionen in Java.

**1. Aufgabe** (8 Punkte)

Mit der Invarianten, mit der wir die partielle Korrektheit des Divisions-Programms aus der Vorlesung gezeigt haben, können wir leider nicht die Terminierung der **while**-Schleife beweisen.

Wir wählen daher mit Blick auf den Terminierungsbeweis $\{divisor > 0\}$ als Invariante.

a) Beweisen Sie zuerst die partielle Korrektheit der Programmformel innerhalb folgender Python **div**-Funktion:

**def** div(dividend, divisor):

$\{P\} \equiv \{dividend \geq 0 \ \wedge \ divisor > 0 \ \}$

result = 0
rest = dividend

$\{INV\} \equiv \{ \ divisor > 0 \ \}$

**while** (rest>=divisor):
    rest = rest - divisor
    result = result + 1
    assert divisor > 0

$\{Q\} \equiv \{(dividend == resutl * divisor + rest) \wedge rest \geq 0 \wedge divisor > rest\}$

**return** (result, rest)

b) Beweisen sie die Terminierung der **while**-Schleife.

**2. Aufgabe** (6 Punkte)

a) Definieren Sie eine **IEEE_Test**-Klasse mit entsprechender **main**-Methode, die folgende Deklarationen beinhaltet.

```
double x = 1.0;
double y = 0.0;
int m = 3;
int n = 1;
int max = 2147483647;
int min = -2147483648;
double z = Double.POSITIVE_INFINITY;
```

b) Berechnen Sie folgende Ausdrücke, geben Sie die Ergebnisse aus und erläutern Sie mit Kommentaren im Programm die Ausgabe.

| | | | | |
|---|---|---|---|---|
| y/x | (x/y+x/y) | (x/y-x/y) | x/m | y/y |
| z/z | n/m | Math.sqrt(-x) | Math.log(-x) | max+1 |
| min-1 | (max+1)==(min) | Double.MAX_VALUE*2 | 3\|4 | ~5 |
| true \|\| 12%3 == 1 | 1<2 && 1<0 | 2^2^3 | true? 8 : 7 | |

**3. Aufgabe** (18 Punkte)
Definieren Sie eine **SomeCalculations**-Klasse in Java, die folgende statische Methoden beinhaltet:

a) Eine **querSumme** Methode, die die Quersumme einer Zahl berechnet und eine **multipleOf3** Methode, die überprüft, dass, wenn die Quersumme durch drei teilbar ist, auch die Zahl selber durch drei teilbar ist, und einen Wahrheitswert zurückgibt.

b) Eine **weekday** statische Methode, die nach Eingabe eines Datums (in Form von drei positiven Zahlen) den Wochentags-Namen mit Hilfe folgender Formeln des Gregorianischen Kalenders berechnet. Die Formeln berechnen eine Zahl zwischen 0 (Sonntag) und 6 (Samstag).

$$y_0 = year - \frac{14 - month}{12}$$

$$x = y_0 + \frac{y_0}{4} - \frac{y_0}{100} + \frac{y_0}{400}$$

$$m_0 = month + 12\left(\frac{14 - month}{12}\right) - 2$$

$$Name = \mathrm{mod}\left(\left(day + x + \frac{(31 \cdot m_0)}{12}\right), 7\right)$$

die **weekday** Methode soll in der **main**-Methode ausgetestet werden, indem kontrolliert wird, dass der Wertebereich der eingegebenen Zahlen korrekt ist und entsprechende Fehlermeldungen ausgibt.

Verwenden Sie für die Eingabe innerhalb der **main** Methode die **Keyboard**-Klasse, die unter *Resources -> Materia*l zu Verfügung gestellt wird.

In der **main**-Methode soll mit Hilfe einer **switch**-Anweisung die Ergebniszahl in einen Wochentag-Namen verwandelt und ausgeben werden.

c) Schauen Sie das Python-Beispiel aus der Vorlesung an, das einen Glücksspieler simuliert und programmieren Sie eine statische **glueckspieler**-Methode, innerhalb der ein Spieler simuliert wird. Die Signatur der Methode soll wie folgt aussehen:

```
public static int gluecksspieler( int bargeld) { ... }
```

Nach jeder Wette soll eine Zeile mit Dollar-Zeichen ausgegeben werden, die den Stand des Restgeldes darstellt.

Die Methode soll als Rückgabewert berechnen, wie lange es dauert (Anzahl der Wetten), bis der Glücksspieler pleite ist.

Verbessern Sie die Methode, sodass der Spieler das Spiel verlässt, wenn er sein Startgeld verdoppelt hat.