Java-Programmierkurs
Prof. Dr. Margarita Esponda
WS 2012/2013

2. Übungsblatt

Ziel: Auseinandersetzung mit **if-else**-, **switch**-Anweisungen, **logischen Ausdrücken** und **while**-Schleifen in Java.

1. Aufgabe (4 Punkte)

Eine Mersenne-Zahl hat die Form $\mathbf{2^n-1}$. Schreiben Sie ein Java-Programm, das bei Eingabe von **n** die entsprechende Mersenne-Zahl berechnet. Für die Eingabe von **n** verwenden Sie die **Keyboard**-Klasse, dessen Eingabe-Methoden in der Vorlesung besprochen worden sind.

2. Aufgabe (6 Punkte)

Wenn die Seitenlängen eines rechtwinkligen Dreiecks **natürliche Zahlen** sind, werden diese Zahlen als **pythagoräische Zahlentripel** bezeichnet.

Definieren Sie eine Klassenmethode **PythagorasTriangle**, die bei Eingabe dreier natürlicher Zahlen feststellen kann, ob es sich um die Seitenlängen eines rechtwinkligen Dreiecks handelt oder nicht. Mit anderen Worten soll die Methode testen, ob die eingegebenen natürlichen Zahlen pythagoräische Zahlentripel sind.

3. Aufgabe (10 Punkte)

Schreiben Sie ein Java-Programm, das nach Eingabe eines Datums (in Form von drei positiven Zahlen) den Wochentag-Namen mit Hilfe folgender Formeln des Gregorianischen Kalenders berechnet. Die Formeln berechnen eine Zahl zwischen 0 (Sonntag) und 6 (Samstag).

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

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

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

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

Das Programm soll am Ende mit Hilfe einer **switch**-Anweisung die Zahl in eine Wochentag-Namen verwandeln und als Ergebnisse ausgeben.

Zwei Bonuspunkte werden vergeben, wenn das Programm richtig kontrolliert, dass der Wertbereich der eingegebenen Zahlen korrekt ist und entsprechende Fehlermeldungen ausgibt.

4. Aufgabe (6 Punkte)

a) Verändern Sie das Java-Programm aus der Vorlesung, das einen Glücksspieler simuliert, indem Sie die Berechnung nicht mehr in der **main**-Methode programmieren, sondern eine statische Glücksspieler-Methode programmieren, innerhalb der die Berechnung gemacht wird. Die Signatur der Methode soll wie folgt aussehen:

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

b) Verändern Sie Ihr Programm, sodass mit Hilfe einer **while**-Schleife nach jeder Wette eine Zeile mit Dollar-Zeichen ausgegeben wird, die den Stand des Restgeldes darstellt.

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

5. Aufgabe (6 Punkte)

a) Kopieren Sie die Klassen **TestTime**, **ClockFrame**, **MyTimer,** **IllegalTimeException**, **Timer** und **Time** aus unserer Webseite (clock-zip);

übersetzen Sie alle Klassen wie folgt:

javac IllegalTimeException

javac Time.java

javac MyTime.java

javac Timer.java

javac ColckFrame.java

javac TestTime.java

und starten Sie die Programmausführung mit der **TestTime**-Klasse.

b) Programmieren Sie den Inhalt der Methoden der Klasse MyTime, die noch leer sind.

```
public class MyTime implements Time {
   ...
      // Methoden
      public void nextHour() {
          // hier programmieren!!
      }
      public void nextMinute() {
          // hier programmieren!!
```

```
        }
        public void nextSecond() {
            // hier programmieren!!
        }
        public void previousHour() {
            // hier programmieren!!
        }
        public void previousMinute() {
            // hier programmieren!!
        }
        public void previousSecond() {
            // hier programmieren!!
        }
    ...
} // end of class Time
```

c) Übersetzen und testen Sie die Methoden der Klasse MyTime in Ihrem Clock-Programm.