P1 - Serie 04

Lukas Batschelet (16-499-733)

Theorieaufgaben

1. Schaltjahre

Aufgabe

Die gregorianische Schalttagsregelung besteht aus folgenden Regeln:

- Die durch 4 ganzzahlig teilbaren Jahre sind, abgesehen von den folgenden Ausnahmen, Schaltjahre.
- Säkularjahre, also die Jahre, die ein Jahrhundert abschliessen (z.B. 1800 oder 1900), sind *keine* Schaltjahre, es sei denn, dass das Säkularjahr auch durch 400 ganzzahlig teilbar ist (zum Beispiel das Jahr 2000).In diesem Fall ist auch das Säkularjahr ein Schaltjahr.

Schreiben Sie eine Methode, die für ein als Parameter übergebenes Jahr `year` überprüft, ob dieses ein Schaltjahr ist oder nicht. Erzeugen Sie eine entsprechen Ausgabe. Geben Sie eine Fehlermeldung aus, falls das Jahr kleiner ist als 1582 (das Jahr in dem der gregorianische Kalender eingeführt wurde).

Mögliche Lösung

```
public boolean isLeapYear(int year) {

final int GREGORIAN_START_YEAR = 1582;

        if (year >= GREGORIAN_START_YEAR) {

                if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {
                        System.out.println("Das Jahr " + year + " ist ein Schaltjahr!");
                        return true;
                } else {
                        System.out.println("Das Jahr " + year + " ist kein Schaltjahr!");
                        return false;
                }

        } else {
                System.out.println("Fehler: Gregorianischer Kalender noch nicht eingeführt");
                return false;
        }
}
```

2. Verschachtelte Vergleiche

Aufgabe

Es sei

(a) `int num1 = 5, num2 = 4;`
(b) `int num1 = 8, num2 = 4;`
(c) `int num1 = 10, num2 = 11;`
(d) `int num1 = 10, num2 = 12;`
(e) `int num1 = 10, num2 = 13;`

Was ist die Ausgabe des folgenden Code Fragments für diese fünf Fälle?

```
if (num1 >= num2) {
        System.out.print("1 ");
        System.out.print("2 ");
        num1 = num2 - 1;
}
System.out.print("3 ");
if ((num1 + 1) >= num2)
        System.out.print("4 ");
else
        if ((num1 + 2) >= num2) {
                System.out.print("5 ");
                System.out.print("6 ");
        } else
                System.out.print("7 ");
System.out.print("8 ");
```

Lösung

```
(a) 1 2 3 4 8
(b) 1 2 3 4 8
(c) 3 4 8
(d) 3 5 6 8
(e) 3 7 8
```

3. Methode `isIsoceles(int a, int b, int c)`

Aufgabe

Schreiben Sie eine Methode `isIsosceles`, die drei ganze Zahlen als Parameter entgegennimmt (die Längen der drei Seiten eines Dreiecks). Die Methode gibt `true` zurück, falls das Dreieck gleichschenklig aber *nicht* gleichseitig ist (also nur dann, wenn *genau zwei* Seiten gleich lang sind).

Lösung

```java
public boolean isIsoceles(int a, int b, int c) {
    return (a == b && a != c) || (a == c && a != b) || (b == c && b != a);
}
```

4. Methode `countA(String name)`

Aufgabe

Schreiben Sie eine Methode `countA`, die in einem als Parameter übergebenem `String` `name` die Anzahl der Zeichen `'a'` und `'A'` zählt und diese Anzahl zurückgibt.

Bemerkung: Lösen mit einer `for`-Schleife

Mögliche Lösung

```java
public int countA(String name) {
        int count = 0;
        for (int i = 0; i < name.length(); i++) {
                char c = name.charAt(i);
                if (c == 'a' || c == 'A') {
                        count++;
                }
        }
        return count;
}
```

5. Ausgabe einer `do`-Schleife

Aufgabe

Welche Ausgabe erzeugt folgendes Code-Fragment?

```java
int low = 0, high = 5;
do {
        System.out.println(low);
        low++;
} while (low < high);
```

Lösung

```
0
1
2
3
4
```

6. Ausgabe einer verschachtelten `for`-Schleife

Aufgabe

Welche Ausgabe erzeugt folgendes Code-Fragment?

```
for (int i = 1; i <= 3; i++) {
        for (int j = i; j > 0; j--)
                System.out.println(i + " " + j);
}
```

```
1 1
2 2
2 1
3 3
3 2
3 1
```

7. Umwandeln einer `for`-Schleife in eine `while`-Schleife

Aufgabe

Schreiben Sie das folgende Code-Fragment mit Hilfe einer `while`-Schleife um.

```
int value = 0;
for (int num = 10; num <= 40; num+=10) {
        value += num;
}
System.out.println(value);
```

```
int value = 0, num = 10;
while (num <= 40) {
value += num;
num += 10;
}
System.out.println(value);
```

Implementationsaufgaben

Aufgabe

Programmieren Sie eine Klasse `Coin`, die eine Münze repräsentieren soll. Eine Münze zeigt entweder Kopf oder Zahl an (speichern Sie diese Information als Variable vom Typ `boolean`).

Der Konstruktor der Klasse `Coin` soll eine Münze zufällig instanziieren – also mit 50% Wahrscheinlichkeit soll eine neu instanziierte Münze Kopf zeigen (und sonst Zahl). Programmieren Sie hierzu eine Methode `flip`, die den Münzwurf simuliert. Ergänzen Sie Ihre Klasse mit einer Methode `equals` und einer Methode `toString`. Zwei Münzen sind gleich, wenn beide die gleiche Seite anzeigen. Die Methode `toString` soll Kopf oder Zahl zurückgeben – verwenden Sie hierzu einen *Conditional Operator*.
Programmieren Sie ein Programm `CoinRace`: Dieses Programm soll zwei `Coin` Objekte solange werfen, bis einer der beiden Münzen dreimal hintereinander Kopf angezeigt hat (Zählen Sie die Runden mit). Diese Münze gewinnt das Spiel.

Geben Sie das Resultat aller Münzwürfe aus und am Schluss geben Sie an, welche der beiden Münzen gewonnen hat aus. Also bspw:

```
Runde 1: Zahl Zahl
Runde 2: Kopf Zahl
Runde 3: Kopf Kopf
Runde 4: Kopf Kopf
Münze 1 gewinnt
```

Hinweis: Es kann auch Unentschieden geben!

Aufgabe

Übernehmen Sie die Klasse `Dice` aus dem Skript. Schreiben Sie dann eine Klasse `PairOfDice`, welche aus zwei `Dice` Objekten besteht. Definieren Sie Methoden zum Setzen und Auslesen der Punkte der einzelnen Würfel, eine Methode um beide Würfel zu werfen und eine Methode, welche die aktuelle Summe der Würfel zurückgibt.

Aufgabe

Implemetieren Sie das Spiel *Pig*. In diesem Spiel gewinnt derjenige Spieler, der zuerst die Punktesumme 100 gesammelt hat. Innerhalb eines Spielzugs wirft der aktuelle Spieler so oft er möchte je zwei Würfel (verwenden Sie also die Klasse `PairOfDice` aus Aufgabe 2). Dabei werden alle Punkte zusammengezählt, bis entweder eine EINS gewürfelt wird, dann ist der Zug zu Ende und alle Punkte dieses Zugs sind verloren und der andere Spieler ist an der Reihe, oder der Spieler die Würfel weiterreicht und seinen Zugfreiwillig beendet. Nur in diesem Fall werden die gewürfelten Augen aufsummiert und dem Konto des Spielers gutgeschrieben. Sollte der Spieler zwei EINSEN gleichzeitig würfeln, verliert er sämtliche bis zu diesem Zeitpunkt gesammelten Punkte auf seinem Konto und sein Zug ist ebenfalls zu Ende.

Nach jedem Wurf muss ein Spieler also entscheiden, ob er weiter würfeln will (und so riskiert, dass er die Punkte des aktuellen Zuges oder sogar sämtliche Punkte verliert) oder ob er die Würfel an den Gegenspieler abgeben möchte (und so riskiert, dass der Gegenspieler gewinnt).

Implementieren Sie das Spiel in den Modi Ein- und Zweispieler: Im Einspielermodus spielt der Spieler gegen den Rechner. Die Strategie des Rechners ist es, solange die aktuelle Summe kleiner ist als 20, weiterzuwürfeln. Bei einer Summe grösser-gleich 20 gibt er den Zug also immer freiwillig ab.