Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Prof. Dr. A.B. Cremers

Aufgabenblatt 1

— Veröffentlicht am 28.04.2010 —

Abgabe bis 04.05.2010, 13:00 per SVN

Aufgabe 1 *(Wahr oder falsch? - 2 Punkte)*

Überlegen Sie sich für jede Aussage ein Argument, mit dem Sie eindeutig bestimmen können, ob die Aussage wahr oder falsch ist. Für jede richtig angekreuzte Antwort werden 0,5 Punkte angerechnet, für jede falsch angekreuzte Antwort 0,5 Punkte abgezogen. Nicht angekreuzte Antworten geben keinen Punktabzug.

wahr	falsch	
		Java-Quellcode-Dateien lassen sich mit Microsoft Word erstellen.
		Alle benötigten Java-Klassen müssen zur Ausführung im Quellcode vorliegen.
		Man kann Java-Quellcode mit speziellen Kommentaren versehen, so dass sich daraus automatisch HTML-Dokumentationen erzeugen lassen.
		Wenn einer Java-Variable vom Typ `int` eine Gleitkommazahl (z.B. *5,7*) zugewiesen wird, dann wird die Zahl automatisch abgerundet.

Aufgabe 2 *(Eingabe und Classpath - 4 Punkte)*

Hinweis: Sie sollten, um die Punkte dieser Aufgabe zu erhalten, das in dieser Aufgabe gelernte Wissen Ihrem Tutor in den Übungen präsentieren können

a) Öffnen Sie einen beliebigen Text-Editor (z.B. den *Windows Editor*) und geben Sie folgendes Programm ein:

```
public class Greeting {
  public static void main(String[] args) {
    System.out.print("Wie ist dein Name? ");
    String name = Input.readString();
    System.out.println("Hallo " + name + "!");
  }
}
```

Speichern Sie die Datei unter dem Namen `Greeting.java`. Öffnen Sie nun eine Kommandozeile[1] und wechseln Sie in das Verzeichnis, in dem Sie die Datei abgespeichert haben. Nun können Sie versuchen, das Programm mit dem Befehl `javac Greeting.java` zu übersetzen.[2]

Dabei sollten Sie die folgende Fehlermeldung erhalten: `Greeting.java:4: cannot find symbol`. Damit will der Compiler Ihnen mitteilen, dass ihm eine Benutzer-Bibliothek zur Übersetzung des Quelltextes fehlt. Die fehlende Bibliothek `input.jar` finden Sie auf der Webseite der Vorlesung.

Speichern Sie die `.jar`-Datei im gleichen Verzeichnis wie die oben erstellte `.java`-Datei und wiederholen Sie die Compilierung, wobei Sie mit Hilfe des zusätzlichen Parameters `-cp` (für *Classpath*)[3] dem Compiler den Pfad zu Ihrer Bibliothek mitteilen: `javac -cp input.jar;. Greeting.java`

Nun können Sie das Programm starten. Dabei müssen Sie aber wiederum den *Classpath* mit angeben: `java -cp input.jar;. Greeting`

[1] Unter Windows können Sie dazu die Windowstaste+R drücken und dann `cmd` eintippen, gefolgt von einem *Return*.

[2] Erhalten Sie dabei die Fehlermeldung, dass das Programm `javac` nicht gefunden wurde, so überprüfen Sie, ob sich das `bin`-Verzeichnis Ihrer JDK-Installation im `PATH` befindet.

[3] Standardmäßig enthält der Classpath nur das aktuelle Verzeichnis. Dies kann durch eine Umgebungsvariable oder eben den Parameter `-cp` überschrieben werden. Dabei ist es möglich, mehrere Bibliotheken und Verzeichnisse anzugeben, die jeweils durch `;` getrennt werden.

b) Versuchen Sie, das Programm nun mit *Eclipse* zu entwickeln. Starten Sie dazu Ihre Eclipse-Installation und erstellen Sie mit *File* → *New* ein neues Java-Projekt. Importieren Sie (z.B. via Drag'n'Drop) die beiden Dateien `Greeting.java` und `input.jar` in das neu erstellte Projekt. Ein rotes Kreuz signalisiert Ihnen, dass die Datei `Greeting.java` einen Fehler enthält. Öffnen Sie die Datei in Eclipse per Doppelklick, so sehen Sie, dass wiederum in Zeile 4 das `Input` bemängelt wird.

Eclipse verwaltet den *Classpath* unabhängig von Parameterangaben in den Projekteinstellungen. Fügen Sie die Datei `input.jar` dem *Classpath* hinzu, indem Sie diese mit der rechten Maustaste anklicken und aus dem Kontextmenü den Befehl *Build Path→Add to Build Path* auswählen. Augenblicklich verschwindet das rote Fehlerkreuz. Starten Sie nun das Programm, indem Sie aus dem Eclipse-Kontextmenü der Datei `Greeting.java` den Befehl *Run As→Java Application* auswählen.

Aufgabe 3 *(Programmierung - 7 Punkte)*

a) Implementieren Sie in der Klasse Exponent eine Funktion zur rekursiven Berechnung von a^n. Der Prototyp der Funktion soll dabei wie folgt aussehen:

```
static public int exponent (int a, int n)
```

Sie können den folgenden Rahmen verwenden:

```
class Exponent {
  static public void main (String[] args){
    if (args.length == 2) {
      System.out.println(args[0] + "^" + args[1] + " = " +
                         exponent(Integer.parseInt(args[0]),
                                  Integer.parseInt(args[1])));
    }
    else {
      System.out.println("Bitte mit java Exponent <a> <n> aufrufen");
    }
  }
  ...
}
```

b) Erweitern Sie ihr Programm `Exponent` dahingehend, dass es dem Benutzer erlaubt, die Werte `a` und `n` während der Ausführung interaktiv einzugeben. Benutzen Sie zum Einlesen der Werte die Funktion `Input.readInt()`, welche ebenfalls durch Einbinden der Datei `input.jar` aus Aufgabe 2 zur Verfügung gestellt wird.

Aufgabe 4 *(Programmierung - 12 Punkte)*

a) Implementieren Sie in Java eine rekursive Funktion, die zu einem Eingabeparameter n den Wert $n! = n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 1$ zurückgibt *(Fakultätsfunktion)*. Ein- und Rückgabeparameter sollen den Typ `long` besitzen.

b) Die für nicht-negative ganze Zahlen n und k definierte Funktion

$$\left(\begin{array}{c} n \\ k \end{array} \right) := \begin{cases} \frac{n!}{k!(n-k)!} & \text{für } 0 \leq k \leq n \\ 0 & \text{für } 0 \leq n < k \end{cases}$$

(gesprochen „n über k“) heißt *Binomialkoeffizient*. Implementieren Sie die Binomialkoeffizienten-Funktion. Die Parameter sollen vom Typ `long` sein, ebenso wie der Rückgabewert.
Bemerkung: Es gilt

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n(n-1)...(n-k+1)}{1 \cdot 2 \cdot ... \cdot k}$$

c) Kombinieren Sie nun die beiden Funktionen zur Lösung des *modifizierten Lottoproblems*: Aus n Zahlen lassen sich – bei Berücksichtigung der Anordnung – k Zahlen ohne Zurücklegen auf

$$\left(\begin{array}{c} n \\ k \end{array} \right) k!$$

Arten auswählen.