Programmierstarthilfe SS 2008

Fakultät für Ingenieurwissenschaften und Informatik

9. Blatt

Für den 23. und 24.6.2008

Organisatorisches

Um auf die Mailingliste aufgenommen zu werden schicke einfach eine Mail an `guido.de-melo@uni-ulm.de`.

Die Webseiten zur Veranstaltung sind unter `http://www.uni-ulm.de/in/mi/lehre/ss2008/programmierstarthilfe.html` zu finden.

ACM Programmierwettbewerb

Am 11. Juli findet wieder der ACM Programmierwettbewerb an der Universität Ulm statt. Der Wettbewerb dient als lokaler Ausscheidungswettbewerb für die Qualifikation zum South West European Regional Contest (SWERC). Wenn ihr Spass am Programmieren habt und vielleicht die langjährige Erfolgsgeschichte Ulmer Programmiermannschaften weiterführen wollt, könnt ihr euch unter `http://www.informatik.uni-ulm.de/acm/Locals/2008/` für den Wettbewerb anmelden. Die Teilnahme ist kostenlos aber auf keine Fall umsonst :)

Weiter Informationen über vergangene Wettbewerbe und Aktivitäten der Programmiermannschaft findet ihr unter `http://www.informatik.uni-ulm.de/acm/`.

1. Gültigkeit von Variablen

Wie ihr sicher schon bemerkt habt, kann man nicht an jeder Stelle auf jede Variable, die irgendwo im Programm deklariert und benutzt wurde, zugreifen. Zum Beispiel wird Java folgendes Programm, das einen Wert in einem Array suchen und die Position zurückgeben soll, nicht kompilieren:

```
1  class meinProgramm{
2          public int find(int[] arr, int wert) {
3                  for (int i=0 ; i<arr.length ; i++){
4                          if (arr[i]==wert) {
5                                  break;
6                          }
7                  }
8                  return i;
9          }
10 }
```

Das Programm durchläuft das Array, bricht durch `break` die Schleife ab, sobald der gesuchte Wert gefunden wurde, und gibt dann den Index `i` zurück, bei dem der Abbruch passiert ist. Wo also liegt der Fehler?

Das Problem ist, dass in Java Variablen immer nur innerhalb eines Blocks gültig sind. Ein Block ist ein Abschnitt der durch geschweifte Klammern eingeschlossen wird. Die Anweisungen innerhalb der for-Schleife oben sind also ein solcher Block. Er steht innerhalb des Blocks der Methode `find`, der wiederum steht in dem Block, der die gesamte Klasse enthält. In jedem Block sind nur die Variablen bekannt, die dort definiert wurden oder in übergeordneten Blöcken stehen. Aufs Beispiel oben bezogen: Die Variable `i` ist im Block der Methode `find` unbekannt, da sie in einem inneren Block, dem der for-Schleife deklariert wird.

Übrigens müssen Blöcke nicht immer dazu da sein Klassen, Methoden, Schleifen oder bedingte Anweisungen einzuschließen. Man kann auch einfach so Gültigkeitsbereiche von Variablen bestimmen und durch geschweifte Klammern einschließen:

```
1  public static void main(String[] args) {
2          {
3                  int a=5;
4                  int b=3*a+6;
5          }
6          {
7                  int a=4;
8                  int b=3*a+6;
9          }
10     }
```

Dadurch kann man Namen doppelt verwenden, allerdings wird dann leicht unübersichtlich, welche Variable gemeint ist.

2. API - Dokumentation

Wie ihr vielleicht selbst schon bei euren ersten Programmierschritten bemerkt hat, benötigt man im Programmieralltag häufig bereits bestehende Methoden, Klassen oder Programmteile. Da es sehr viele Probleme oder Aufgaben gibt, die bei der Programmierung häufig auftreten und gelöst werden müssen, gibt es bei Java bereits eine große Anzahl von fertigen Klassen. Da die Existenz der Klassen allein aber noch nicht ausreicht, gibt es außerdem dazu eine ausführliche Dokumentation dazu. Dort werden die einzelnen Pakete, Klassen und Methoden vorgestellt und ihre Verwendung erklärt. Als künftige(r) Java-ExpertIn gehört es mit zum Grundwissen, mit dieser API-Dokumentation umgehen zu können und dort Informationen zu finden.

Auch bei der Entwicklung von größeren Softwareprojekten später ist es notwendig, mit der Dokumentation von fremden Klassen und Methoden umgehen zu können. Da sich solche Do-

kumentationsübersichten automatisch aus beliebigen Java-Code erzeugen lassen, wenn dort die Kommentare in einem bestimmten Stil formatiert sind, werden sie häufig auch zur Dokumentation von Firmensoftware, freien Projekten, etc. verwendet.

Auf der linken Seite der Dokumentation ist eine Liste der Klasse zu finden, auf der Hauptseite in der Mitte gibt es dann jeweils Informationen zu den einzelnen Klassen. Neben einer textuellen Beschreibung der Klasse und Informationen über geerbte Attribute und Methoden gibt es außerdem eine Übersicht der Konstruktoren, Methoden und Konstanten jeder Klasse.

Die aktuelle API-Dokumentation zu Java findet ihr hier: `http://java.sun.com/javase/6/docs/api/`

3. Aufgaben

3.1. Wettbüro

Du sollst ein Programm für ein Online-Wettbüro schreiben, das Fußballwetten annimmt. Dabei gibt es pro Spiel verschiedene Wettmodi: Man kann auf den Sieg einer Mannschaft (bzw. ein Unentschieden) setzen oder zusätzlich das Ergebnis tippen. Nach dem Spiel werden die gesetzten Beträge nach einer vorher gesetzten Quote (jeweils für Sieg, Niederlage, Unentschieden, richtig getippten Ergebnis und richtiger Tordifferenz) ausgezahlt.

Z.B. kann für das Spiel Litauen gegen Deutschland die Quote für den Sieg der Deutschen bei 1.2 liegen, für den Sieg der Litauer bei 20.3, für ein Unentschieden bei 15.0. Ein Tipper, der 10 Euro auf Litauen gesetzt hat, bekommt bei einem Sieg der Litauer $20.3 * 10 = 203$ Euro ausgezahlt. Für ein richtiges getipptes Ergebnis kann es eine Quote von 10.0 geben, für die richtige Tordifferenz von 5.0. Der entsprechende Betrag wird zusätzlich ausgezahlt.

a) Zunächst soll es eine Klasse `Wette` geben. In dieser Klasse wird der Wettmodus z.B. als Integer abgespeichert, außerdem wird Name des Tippenden, der gesetzte Betrag und der Tipp abgespeichert. Der Tipp sieht folgendermaßen aus: Im ersten Wettmodus ist es ein Integer $\in \{0, 1, 2\}$ für Unentschieden oder den Sieg der Mannschaft 1 oder 2, im zweiten Wettmodus zwei Integer mit den entsprechenden Toranzahlen. Stelle zwei entsprechende Konstruktoren zur Verfügung. Wird nicht das Ergebnis getippt, werden die entsprechenden Variablen auf -1 gesetzt.

b) Lege eine Klasse `Spiel` an, deren Objekte jeweils für ein Spiel die Wetten entgegennehmen. Im Konstruktor sollen die Quoten und die Namen der beiden Mannschaften übergeben werden. Außerdem soll es ein Array für Objekte der Klasse `Wette` geben.

c) Nun soll es in der Klasse `Spiel` zwei Methoden `tipp` geben. Einer wird entsprechend dem ersten Wettmodus nur ein Integer übergeben, der anderen zwei. Außerdem werden der Name des Tippenden und der gesetzte Betragen übergeben. In der Methode soll ein Objekt der Klasse `Wette` angelegt und im Array abgespeichert werden.

d) Implementieren in der Klasse `Spiel` nun noch die Methode `auszahlung`, der das Endergebnis des Spiels übergeben wird. In der Methode soll das Wettarray durchlaufen und entprechend der Quoten für jeden Tippenden der gewonnene Betrag ausgegeben werden.

e) Teste dein Programm, indem du ein oder zwei Objekte der Klasse `Wette` anlegst, jeweils einige Tipps eingibst und dann die Methode `auszahlung` aufrufst.

3.2. Wo gilt was?

Schau dir den folgenden Programmcode an und überlege dir (hier wird kein PC benötigt!), wann welche Variable gilt. Welche Ausgaben werden an den vorgesehenen Stellen getätigt. Gibt es Dinge im Programm, die so nicht funktionieren? Warum?

```
1  class Ding {
2      public int a = 0;
3      public String toString() {
4          return ""+a;
5      }
6  }
7
8  public class Test {
9      public static String methode(double b, Ding a) {
10         String c = "Methoden-String";
11         a.a = 12;
12         System.out.println(a+"\t"+b+"\t"+c);
13         return c+a;
14     }
15     public static void main(String ... args) {
16         {
17             int a = 0; String b = ""; double c = 1.0;
18             System.out.println(a+"\t"+b+"\t"+c);
19         }
20         double a = 2.4;
21         for(int b=0; b<a; b++) {
22             System.out.println(a+"\t"+b+"\t"+c);
23         }
24         System.out.println(a+"\t"+b+"\t"+c);
25         Ding b = new Ding();
26         String c = methode(a,b);
27         System.out.println(a+"\t"+b+"\t"+c);
28     }
29 }
```

3.3. Funktionen für Strings aus der API

Öffne die API-Dokumentation (`http://java.sun.com/javase/6/docs/api/`) und suche die Klasse String. Suche nun zu den folgenden Problemem eine passende Methode heraus und teste sie anhand eines kleinen Testfalls. Wichtig: Es soll keine Methode hierbei selbst implementiert werden, beschränke dich auf die bereits existierenden Methoden aus der API.

a) Wie kann man prüfen, ob ein String leer ist?

b) Wie bekommt man die Länge eines Strings?

c) Wie entfernt man überflüßige Leerzeichen an Anfang und Ende eines Strings?

d) Wie wandelt man einen String in eine Text, der nur aus Großbuchstaben besteht, um?

e) Wie testet man, ob ein String mit einer bestimmten Zeichenkette endet?

f) Wie vergleicht man einen String mit einem zweiten, ohne dabei auch Klein-/Großschreibung zu achten?

g) Wie bekommt man den Character eines Strings an der 3. Stelle?

h) Wie findet man die Stelle des letzen Vorkommens eines Character in einem String?

i) Wie ersetzt man in einem String alle Vorkommen einer Zeichenkette durch eine andere Zeichenkette?

j) Angenommen, ein gegebener String würde einen Satz enthalten. Wie bekommt man am einfachsten ein Array der darin enthaltenen Wörter? Überlege dir zunächst, was die Wörter voneinander trennt.

3.4. Klasse zur Generierung von Passwörtern

Es soll eine Klasse `PasswordGenerator` zur Erzeugung von zufälligen Passwörtern geschrieben werden. Der Konstruktor dieser Klasse soll drei `boolean`-Werte übergeben bekommen, die angeben, ob kleine Buchstaben, große Buchstaben sowie Ziffern vorkommen dürfen. Hierbei muss mindestens eine Zeichenart aktiviert sein, ansonsten sollen standardmäßig zumindest die kleinen Buchstaben verwendet werden. Eine Methode `generatePassword(int length)` soll anschließend gemäß den beim Konstruieren des Objektes gültigen Zeichenklassen entsprechende Passwörter zufällig erzeugen und diese als String zurückgeben.

```
1 // PasswordGenerator-Objekt erzeugen mit
2 // kleinen und grossen Buchstaben
3 PasswordGenerator meinPasswortMacher = new
4   PasswordGenerator(true,true,false);
5 // Passwort der Laenge 8 erzeugen
6 String passwort1 = meinPasswortMacher.generatePassword(8);
```

a) Für diese Klasse benötigst du Zufallswerte. Bisher haben wir dafür die Methode `Math.random()` genutzt. Dieses Mal soll jedoch die fertige Java-Standardklasse Random benutzt werden. Informiere dich in der API-Dokumentation im Internet über ihre Verwendung und ihre Methoden. Überlege dir anschließend, wie du die Klasse sinnvoll für dein Problem benutzen kannst.

   Suche im Internet nach „java api class random" oder folge dem Link:
   `http://java.sun.com/javase/6/docs/api/java/util/Random.html`

b) Implementiere die Klasse `PasswordGenerator` und teste sie. Achte hierbei auf die verschiedenen Zeichenarten, die man - auch kombiniert - auswählen kann.

c) Erzeuge eine weitere Methode `generatePasswordArray(int passwordLength, int passwords)`, die ein Array von Strings mit erzeugten Passwörtern zurückgeben soll. Hierbei soll die Länge sowie die Anzahl der zu erzeugenden Passwörter übergeben werden.

d)** Oft sind zufällige Passwörter sehr schwer zu erraten, aber auch kaum zu merken. Aus diesem Grund ist es hilfreich festzulegen, dass sich bei Buchstaben im Passwort immer Vokale und Konsonanten abwechseln, um das damit erzeugte Passwort aussprechbar zu machen. Schaffst du es, trotzdem auch noch Ziffern unterzumischen, falls diese aktiviert wurden?

3.5. Die Klasse System

Schauen wir uns nun die API-Dokumentation der Klasse System an. Wie kann man das folgende Programm damit implementieren?

a) Zu Beginn soll eine Begrüßung ausgegeben werden. Was hat das mit der Klasse System zu tun?

b) Dann soll ein Zeilenumbruch ausgeführt werden. Beachte: Bei Linux erzeugt das Zeichen line-feed (`\n`) einen Zeilenumbruch, bei MacOS das Zeichen carriage-return (`\r`) und bei Windows die Kombination beider Zeichen (`\r\n`). Wir wollen, dass das Programm überall funktioniert. Tipp: Die Systemvariable für das systemspezifische Zeilenumbruchzeichen heißt `line.separator`.

c) Lass nun den Garbage-Collector laufen und informiere dich dazu, was dieser tut.

d) Miss die Zeit, die der Garbage-Collector benötigte. Das geht am einfachsten, indem du vorher und nachher die Systemzeit speicherst und die Differenz bildest.

e) Wenn der Garbage-Collector zu lange brauchte (nimm dir da eine beliebige Millisekundenzahl), dann soll eine Fehlermeldung ausgegeben werden. Fehlermeldungen schreibt man als guter Programmierer nicht auf die Standardausgabe, sondern auf die Standard-Fehler-Ausgabe.

f) Wenn kein Fehler auftrat soll das Programm normal enden, wenn doch soll es mit Exit-Status -1 enden. Der Fehlercode signalisiert anderen Programmen, dem Benutzer und dem System in fortgeschrittenere Anwendung, ob dein Programm fehlerfrei lief.

4. Bonusaufgaben