Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Christian Böhm
Annahita Oswald, Bianca Wackersreuther

München, 27.11.2009

Einführung in die Programmierung

WS 2009/10

Übungsblatt 6: Arrays, Kontrollstrukturen

Besprechung: 07./09./10./11.12.2009

Ende der Abgabefrist: Montag, 07.12.2009 10:00 Uhr.

Hinweise zur Abgabe:
Geben Sie bitte Ihre gesammelten Lösungen zu diesem Übungsblatt in einer Datei `loesung06.zip` unter `http://www.pst.ifi.lmu.de/uniworx/` ab.
Bitte beachten Sie, dass Aufgabe 6-3 nicht in die Bonusregelung eingeht. Bereiten Sie diese aber bitte trotzdem vor, damit Sie der Übung optimal folgen können.

Aufgabe 6-1 *Arrays* **10 Punkte**

(a) Schreiben Sie in einer Datei `Minimum.java` eine Methode `int minimum(int[] a)`, die für ein Array vom Typ `int[]` diejenige Zahl zurückgibt, die den kleinsten Wert hat.

(b) Erweitern Sie Ihr Programm um eine entsprechende `main`-Methode, so dass Sie Ihre implementierte Methode `int minimum(int[] a)` für die folgenden Beispiele testen können.

- Das Minimum von [14, 79, 142, 99] ist 14.
- Das Minimum von [10, 9, 8, 7, 6, 5, 4, 3, 2, 1] ist 1.
- Das Minimum von [8, 15, -3, 3] ist -3;

Hinweis: Als Hilfestellung können Sie die von uns vorgegebene Datei `Minimum.java` verwenden.

Aufgabe 6-2 *Terminierung von Schleifen* **10 Punkte**

Welche der folgenden Schleifen terminieren, welche terminieren nicht? Begründen Sie Ihre Antworten kurz. Geben Sie Ihre Lösung in einer Datei `schleifen.txt` ab.

(a)
```
int i = 1;
int j = 1;
do
{
  i = i + j;
  j++;
}while(i < 200);
```

(b)
```
int i = 1;
int j = 20;
while(i + j > i)
{
  i = i + 2;
  j--;
}
```

(c)
```
int i = 100;
int j = 27;
while (i != j)
{
  i = i / 20;
  j = j / 3;
}
```

(d)
```
int i = 26;
int j = 24;
for (int x = 0; x < 1000; x++)
{
  i = i / 12 + 23 * x;
  j = (x--) + j + 5;
}
```

Aufgabe 6-3 *Arrays und Kontrollstrukturen* **0 Punkte**

Erstellen Sie eine Datei `SortierteArrays.java`, die eine Methode `int[] sortiere(int[] a, int[] b)` enthält. Dieser Methode werden zwei aufsteigend sortierte `int`-Arrays als Parameter übergeben und liefert ein `int`-Array zurück, das alle Elemente der beiden Arrays `a` und `b` ebenfalls aufsteigend sortiert enthält.

Beispiel:
Gegeben sind die Arrays `int[] a = {-1,1,3,7};` und `int[] b = {1,2,4,6,8};`.
Der Aufruf `merge(a,b);` gibt das Array `{-1,1,1,2,3,4,6,7,8}` zurück.

Sie können annehmen, dass die der Methode übergebenen Arrays aufsteigend sortiert sind.