Prof. Dr. Volker Linnemann
Dr. Sven Groppe
Universität zu Lübeck
Institut für Informationssysteme

Lübeck, den 9.11.2007

Vorlesung „Programmieren“

Wintersemester 2007/2008

3. Übungsblatt

1. Sortierung von Arrays (Programmieraufgabe) (10 Punkte)

Gegeben sei folgende Klasse MinSort, die Sie um zwei Methoden ergänzen sollen.

```
/**
 * Class MinSort sorts the elements of an array.
 *
 * @author (your name)
 */
public class MinSort
{
    // instance variables - replace the example below with your own
    private int[] x;

    /**
     * Constructor for objects of class MinSort
     */
    public MinSort()
    {
        // initialise instance variables
        x=new int[10];
        x[0] = 0;
        x[1] = 526;
        x[2] = 40;
        x[3] = 670;
        x[4] = 1000;
        x[5] = 50;
        x[6] = 66;
        x[7] = -1;
        x[8] = 8;
        x[9] = 780;
    }

    /**
     * returns the minimum in the array x
     * and deletes the minimum in the array x
     *
     * @return     minimum in the array x
     */
    public int getMin()
    {
       // please add your code here…
    }

    /**
     * returns an array, which contains all elements of x
     * in sorted order
     *
     * @return     sorted array x
     */
    public int[] sort()
```

```
    {
      // please add your code here…
    }
}
```

Diese Klasse `MinSort` erzeugt im Konstruktor ein neues Feld mit Feldelementtyp `int` und initialisiert dieses Feld.

Aufgaben:

a) Implementieren Sie die Methode `int getMin()` der Klasse `MinSort`, die das kleinste Element im Feld `x` zurückgibt und dieses kleinste Element im Feld `x` löscht, in dem ein Feld mit einer gegenüber `x` um 1 erniedrigten Feldlänge erzeugt wird, alle Elemente bis auf das kleinste Element aus `x` in dieses neue Feld kopiert werden und `x` auf dieses neu erzeugte Feld gesetzt wird. (5 Punkte)

b) Implementieren Sie die Methode `int[] sort()` der Klasse `MinSort`, die ein Feld mit den Elementen aus `x` zurückgibt, deren Elemente sortiert sind. (5 Punkte)

Bemerkungen:

- Bei der Implementation der Methode `int[] sort()` der Klasse `MinSort` kann die Methode `int getMin()` verwendet werden.
- Es gibt wesentlich effizientere Sortieralgorithmen, die Sie in weiterführenden Veranstaltungen kennen lernen werden. Bei dieser Aufgabe geht es im Wesentlichen um den Umgang und die Verwendung von Feldern, nicht so sehr um die Implementierung eines *effizienten* Sortieralgorithmus.
- Dieses ist eine Programmieraufgabe. Bitte geben Sie die Lösung *sowohl* auf **Papier** *als auch* den Java-Quelltext in elektronischer Form per **Email** an ihren Tutor ab.

2. Programmierung eines Würfelspiels (Programmieraufgabe) (5 Punkte)

Ein Casino bietet das folgende Würfelspiel an. Dabei setzt ein Spieler 1 Euro und würfelt anschließend mit zwei Würfeln. Ob und wie viel er gewonnen hat entscheidet die Augensumme entsprechend der folgenden Tabelle.

Augensumme	**Auszahlung**
2,3,4,5	nichts
6,7,8,9	einfacher Einsatz
10	doppelter Einsatz
11	vierfacher Einsatz
12	sechsfacher Einsatz

Demnach verliert z.B. der Spieler bei einer geworfenen 5 die eingesetzten 1 Euro an die Bank. Bei einer erwürfelten 11 hingegen würde die Bank 4 Euro an den Spieler zurückzahlen; er gewinnt also netto 3 Euro.

Die folgende Java-Methode `roll1W6()` mit der Signatur `int roll1W6( )` sei gegeben, welche einen zufälligen Wert zwischen 1 und 6 zurückliefert:

```
/**
 * Liefert das Ergebnis eines einzelnen Wuerfelwurfes zurueck.
 *
 * @return eine zufaellige Augenzahl eines Wuerfels
 */
 public int roll1W6( ) {
  return (int)(6 * Math.random()) + 1;
 }
```

Aufgaben:

a) Nutzen Sie `roll1W6()`, um eine Methode `roll2W6()` mit der Signatur `int roll2W6( )` zu schreiben, welche die Augensumme von zwei Würfelwürfen zurückliefert.

b) Schreiben Sie eine Methode `outcomeOfRoll()` mit der Signatur `int outcomeOfRoll( int roll )`, welche zu einem Würfelergebnis zwischen 2 und 12 den Gewinn (bzw. Verlust) in Euro bei dem Einsatz von 1 Euro zurückliefert.

c) Nutzen Sie diese Methoden, um mit Hilfe einer weiteren Methode `simulation()` herauszufinden, ob das Spiel die Bank oder den Spieler bevorteilt. Simulieren Sie dazu 100.000 solcher Spiele und summieren Sie den Gewinn bzw. Verlust jeder Runde. Wir sagen, das Spiel bevorteilt den Spieler, wenn dieser nach den 100.000 Runden mit mehr Geld als vorher dasteht, bzw. es bevorteilt die Bank, wenn der Spieler hinterher weniger Geld besitzt. Lassen Sie sich das Ergebnis „Das Spiel bevorteilt ??? mit insgesamt EUR ??? in ??? Runden." oder „Unentschieden nach ??? Runden." als `String` zurückgeben (wobei die ??? im Ergebnis natürlich durch die entsprechenden Werte zu ersetzen sind).

Bemerkungen:

- Dieses ist eine Programmieraufgabe. Bitte geben Sie die Lösung *sowohl* auf **Papier** *als auch* den Java-Quelltext in elektronischer Form per **Email** an ihren Tutor ab.

Bemerkungen:

- Jede Seite soll oben rechts den Namen der Abgebenden und die Übungsgruppennummer (wichtig!) enthalten.
- Lösungen für die Übungsaufgaben sind (in der Regel) zu zweit abzugeben.
- Kommentieren Sie Ihre Lösungen! Besteht eine Lösung aus mehreren Zetteln, so sind diese zusammen zu heften. Bitte keine Hüllen, Mappen, o.ä..
- Bitte schicken Sie *Programmieraufgaben zusätzlich zur Abgabe auf Papier in elektronischer Form per Email* an ihren jeweiligen Tutor.
- Kommentieren Sie ihren Quelltext bei Programmieraufgaben. Dabei sollen keine Trivialitäten kommentiert werden, also bitte keine Kommentare wie

  ```
  x=5; // Wir weisen nun der Variablen x den Wert 5 zu
  ```

  sondern sinnvolle Kommentare, die Ideen des Quelltextabschnittes beschreiben oder auf Unteraufgaben (z. B. a), b), …) hinweisen.
- **Hinreichende Bedingung für die Zulassung zur Klausur:** 50% der erreichbaren Punkte bei jedem Übungszettel (bis auf zwei) und einmaliges Vorrechnen in der Übung
- **Zertifikatskriterium:** Das Bestehen der Klausur am Ende des Semesters

Abgabetermin: Freitag, 16.11.2007, nach der Vorlesung