Programmbeispiele zum Teil II der Vorlesung

Grundlagen der Programmierung

Prof. K.Bothe

WS 15/16


```
// *** II.2 Compilation

class Hello {

  public static void main (String[] args) {

    System.out.println("Hello!");
  }
}
```

```
// *** II.3 Grundlegende Sprachkonstruktionen imperativer Programme

class Temperature {
 //  Convert temperature
 //   from Fahrenheit to Centigrade


  public static void main (String[] args) {
    double tempFahr;  // Fahrenheit
    double tempCels;  // Celsius

    System.out.print("Please type the temperature (deg F): ");
    tempFahr = Keyboard.readDouble();
    tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;
    System.out.print(tempFahr);
    System.out.print(" deg F is ");
    System.out.print(tempCels);
    System.out.println(" deg C");
  }
}
```



```
// *** II.3 Grundlegende Sprachkonstruktionen imperativer Programme

import java.io.*;

class Keyboard {
// Primitive Keyboard input of integers, reals,
// strings, and characters.

static boolean iseof = false;
static char c;
static int i;
static double d;
static String s;

/* WARNING:  THE BUFFER VALUE IS SET TO 1 HERE TO OVERCOME
** A KNOWN BUG IN WIN95 (WITH JDK 1.1.3 ONWARDS)
*/
static BufferedReader input
       = new BufferedReader (
              new InputStreamReader(System.in),1);

  public static int readInt () {
    if (iseof) return 0;
    System.out.flush();
    try {
      s = input.readLine();
    }
    catch (IOException e) {
      System.exit(-1);
    }
    if (s==null) {
      iseof=true;
      return 0;
    }
    i = new Integer(s.trim()).intValue();
    return i;
  }

  public static char readChar () {
    if (iseof) return (char)0;
    System.out.flush();
    try {
      i = input.read();
    }
    catch (IOException e) {
      System.exit(-1);
    }
    if (i == -1) {
      iseof=true;
      return (char)0;
    }
    return (char)i;
  }

  public static double readDouble () {
    if (iseof) return 0.0;
    System.out.flush();
    try {
      s = input.readLine();
```

```
    }
    catch (IOException e) {
      System.exit(-1);
    }
    if (s==null) {
      iseof=true;
      return 0.0;
    }
    d = new Double(s.trim()).doubleValue();
    return d;
  }

  public static String readString () {
    if (iseof) return null;
    System.out.flush();
    try {
      s=input.readLine();
    }
    catch (IOException e) {
      System.exit(-1);
    }
    if (s==null) {
      iseof=true;
      return null;
    }
    return s;
  }

  public static boolean eof () {
    return iseof;
  }

}
```



```
// *** II.6 Iteration (Zyklen, Schleifen)

class TemperatureTable {
// Tabelle mit C/F Temperaturen

 public static void main (String[] args) {
   final double
     LOW_TEMP = -10.0,
     HIGH_TEMP = 10.0;

   double
     cent,  // Grad Celsius
     fahr;  // Grad Fahrenheit

   System.out.println("\tGrad C\t\tGrad F");
   cent = LOW_TEMP;
   while (cent <= HIGH_TEMP) {
     fahr = (9.0/5.0) * cent + 32.0; // C -> F
     System.out.println(
         "\t" + cent + "\t\t" + fahr);
     cent = cent + 1.0;
   }
  }
}
```

```
// *** II.7 Methoden (Algorithmen, Funktionen, ...)

class ZeitPlan {

  private static int hour, minute; // die aktuelle Zeit !

  private static void addMinutes (int m) {

    // erheoht die aktuelle Zeit um m Minuten

    int totalMinutes = (60*hour + minute + m) % (24*60);
    if (totalMinutes < 0)
      totalMinutes = totalMinutes + 24*60;
     hour = totalMinutes/60; minute = totalMinutes%60;
  }

  private static void printTime () {

    // druckt die aktuelle Zeit nach
    // englischen Konventionen: AM, PM, noon, midnight

    if ((hour == 0) && (minute == 0))
      System.out.print("midnight");
    else if ((hour == 12) && (minute == 0))
      System.out.print("noon ");
    else {
      if      (hour == 0) System.out.print(12);
      else if (hour > 12) System.out.print(hour-12);
      else                System.out.print(hour);

      if (minute < 10) System.out.print(":0" + minute);
      else             System.out.print(":" + minute);

      if (hour < 12) System.out.print("AM");
      else           System.out.print("PM");
    }
  }


  private static void printTimeInMinutes () {

     // druckt aktuelle Zeit mit Entsprechung in Minuten

    printTime ();
    System.out.println(" =" + timeInMinutes() + ". Minute des Tages ");
  }


  private static int timeInMinutes () {

    // ermittelt die Anzahl von Minuten seit 0:00 Uhr,
    // die der aktuellen Zeit entspricht

    int totalMinutes = (60*hour + minute) % (24*60);
    if (totalMinutes < 0)
      totalMinutes = totalMinutes + 24*60;
    return totalMinutes;
  }
```



```
  private static void includeNewEntry (int intervalInMinutes, String event) {

    // druckt eine Zeile: Zeitangabe  Veranstaltung;
    // erhoeht Zeit um Laenge (intervalInMinutes) der Veranstaltung

    printTime();
    System.out.print(" ");
    System.out.println(event);
    addMinutes(intervalInMinutes);
  }

  public static void main (String[] args) {

    hour = 8; minute = 30;

    // Druck der Zeitangaben als Plan
    System.out.println("Terminkalender: Zeitangabe + Text");
    System.out.println("-------------------------------");

    includeNewEntry(90, "V PI1");
    includeNewEntry(15, "Pause");
    includeNewEntry(90, "V ThI1");
    includeNewEntry(30, "Pause");
    includeNewEntry(90, "U PI1");
    System.out.println("letzte (aktuelle) Tageszeit in Minuten: ");
    printTimeInMinutes ();
  }
}
```

```
// *** II.7 Methoden (Algorithmen, Funktionen, ...)

class Fakultaet {

 public static int fakultaet (int n) {
    int x, fak=1;

    for (x=1; x<=n; x++)
      fak = fak * x;

    return fak;
 }

 public static void main (String[] args) {

  int x, y;

  x = fakultaet(3);
  System.out.println("3! = " + x);
  y = fakultaet(5);
  System.out.println("5! = " + y);
 }
}
```



```
// *** II.8 Ausdruecke

// Ausgabe der Unicodezeichen 0020 - 00FF im Console-Fenster
// Windows: Ausgabe von Unicode-Zeichen im Console-Fenster erfordert die Code-
//          page 1252. Erfolgt durch das Kommando: 'chcp 1252' im DOS-Fenster.
//          Zusaetzlich im DOS-Fenster die Schriftart 'Lucida Console' waehlen.

import java.awt.*;

class Unicode {

 public static void main (String [] args) {

   for (char  code = '\u0020' ;  code <= '\u00FF' ;  code = (char)(code + 1)) {
     String fill = "";
     if (code < '\u0064')
       fill = " ";                                  // zweistellige Dezimalzahlen re
chtsbuendig

     if (code >= '\u007F' && code <= '\u009F')
        // '?' fuer nichtdruckbares Zeichen
        System.out.print
        (fill + Integer.toString(code) + " " + '?' + " ");
     else
        System.out.print
        (fill + Integer.toString(code) + " " + Character.toString(code) + " ");

     if (code%10 == 0)
       System.out.println();
   }
   System.out.println();
 }
}
```

```
// *** II.10 Eindimensionale Felder

public class Primzahlen {
//   Aufgabe: ermittle Primzahlen bis zu einer Grenze.
//   Technik: Sieb des Eratosthenes
//   Grundidee: Streiche alle Vielfachen von bereits als
//      Primzahl erkannten Zahlen.

  public static void main (String argv[]) {

    boolean[] sieb;         // Position i entspricht Zahl i
                            // sieb[i] = true <-> i ist Primzahl
    int i, j, n;

    System.out.print("Primzahlgrenze: ");
    n = Keyboard.readInt();
    sieb = new boolean[n];  // jetzt erst: Speicherplatz anfordern

    for (i = 2; i < n; i++) sieb[i] = true;   // alle potientiell Primzahl

    for (i = 2; i < n; i++)
        if (sieb[i]) {                        // falls i Primzahl,
            for (j = i+i; j < n; j += i)      // streiche alle Vielfachen
                sieb[j] = false;              // von i aus der Liste:
        }                                     // 2 * i, 3 * i ...

    // hier ergaenzen: Ausgabe der Primzahlen
  }
}
```



```
// *** II.10 Eindimensionale Felder

public class Echo {

  public static void main(String args[]) {

    for (int i=0; i < args.length; i++)
      System.out.print(args[i] + " ");

    System.out.print("\n");
  }
}
```

```
// *** II.10 Eindimensionale Felder

class Monate {
// konstante Arrays +
// Technik 'paralleler' Arrays

  public final static String[]
     MONTH_NAME = {"",
            "Januar",    "Februar",   "Maerz",     "April",
            "Mai",       "Juni",      "Juli",      "August",
            "September", "Oktober",   "November",  "Dezember"};

  public final static int[]
     DAYS_OF_MONTH = {0,
                      31, 28, 31, 30,
                      31, 30, 31, 30,
                      30, 31, 30, 31};

  public final static int
     JANUAR=1,    FEBRUAR=2,  MAERZ=3,     APRIL=4,
     MAI=5,       JUNI=6,     JULI=7,      AUGUST=8,
     SEPTEMBER=9, OKTOBER=10, NOVEMBER=11, DEZEMBER=12;

  static int M = MAI;

  public static void main (String[] args) {
     System.out.println("Monat " + MONTH_NAME[M]
            + " hat " + DAYS_OF_MONTH[M] + " Tage");
  }
}
```



```
// *** II.11 Rekursion, Komplexitaet von Algorithmen

class Power1 {

static int power (int k, int n) {
// Raise k to the power n.
  if (n == 0)
    return 1;
  else {
    int t = power(k, n/2);
    if ((n % 2) == 0)
      return t*t;
    else
      return k*t*t;
  }
}

public static void main (String[] args) {
  int z;
  int i;
  while (true) {
    System.out.print("Enter integer: ");
    z=Keyboard.readInt();
    System.out.print(
          "Enter exponent (integer >= 0): ");
    i=Keyboard.readInt();
    System.out.println(z+"**"+i+" = "+power(z,i));
  }
}
}
```

```
// *** II.11 Rekursion, Komplexitaet von Algorithmen

public class Hanoi {

  static void bewege (
    int n,              // Anzahl der Scheiben 'n'
    char start,         // liegen auf 'start'-Platz
    char hilfe,         // (mithilfe des 'hilfe'-Platzes)
    char ziel) {        // muessen auf 'ziel'-Platz

    if (n == 1)
        System.out.println(" von " + start + " nach " + ziel);
    else {
        bewege(n-1,start, ziel, hilfe);
        System.out.println(" von " + start + " nach " + ziel);
        bewege(n-1,hilfe, start, ziel);
    }
  }

  public static void main (String argv[]) {
    int n;

    System.out.print("Anzahl der Scheiben: ");
    n = Keyboard.readInt();
    if (n > 0) {
      System.out.println("Scheibenbewegungen: ");
      bewege(n,'A','B','C');
    }
    else
      System.out.println("Zahl nicht positiv");
  }
}
```



```
// *** II.12 Such- und Sortierverfahren mit Arrays

public class Quicksort {

  public static void quicksort (int[] a, int links, int rechts) {
    int help ;
    int i = links;
    int j = rechts;
    int x = a[(links+rechts) / 2];   // Vergleichselement

    do {
        while (a[i] < x) i++;
        while (a[j] > x) j--;
        if ( i<=j )  {
            help  = a[i];
            a[i] = a[j];
            a[j] = help;
            i++;
            j--;
        }
    } while (i <= j);
      // Elemente im linken Teil sind kleiner
      // als Elemente im rechten Teil

      // sortiere linken und rechten Teil einzeln,
      // falls mehr als ein Element vorhanden
    if (links < j)    quicksort(a, links, j);
    if (i < rechts )  quicksort(a, i, rechts );
  }


  public static void main (String argv[]) {

    int[] a;
    int n;

       // Groesse des Feldes eingeben
    System.out.print("Groesse des Feldes: ");
    n = Keyboard.readInt();
    a = new int[n];

       // Inhalt des Feldes eingeben:
    System.out.println(n + " Zahlen eingeben: ");
    for (int i = 0; i < n; i++) {
      System.out.print("naechste Zahl:");
      a[i] = Keyboard.readInt();
    }
      // zu Beginn: das gesamte Feld ist zu sortieren
    quicksort(a, 0, n-1);

      // Ausgabe des sortierten Feldes:
    System.out.println("sortiertes Feld: ");
    for (int i=0; i<a.length; i++)
       System.out.print(" "+a[i]);
    System.out.println();
  }
}
```

```
// *** II.12 Such- und Sortierverfahren mit Arrays

public static void lineareSuche
                    (int [] a, int x) {

 for (int i=0;
      (i < a.length) && (x != a[i]);
      i++);
 if (i==a.length)
   System.out.println("Nicht gefunden");
 else
   System.out.println(
     "Gefunden an Position " + i);
}


public static void binaereSuche
                    (int [] a, int x) {
 int links, rechts, mitte;

    links = 0;
    rechts = a.length-1;
    while (links <= rechts) {
        mitte = (links + rechts) / 2;
        if (a[mitte] == x) {
          System.out.println
           ("Gefunden an Position "
                              + mitte);
          return;
        }
        if (a[mitte] < x)
          links  = mitte+1;
        else
          rechts = mitte-1;
    }
    System.out.println("Nicht gefunden");
}
```



```
// *** II.12 Such- und Sortierverfahren mit Arrays

public static int[]
     merge (int[]a, int[]b, n) {
// Mischen von zwei sortierten Folgen
// Resultatwert (return): neue Folge

  int i=0, j=0, k=0;
  int[] c =
     new int[a.length + b.length];

  // mischen, bis ein Array leer
  while ((i<a.length) && (j<b.length)) {
    if (a[i] < b[j])
        c[k++] = a[i++];
    else
        c[k++] = b[j++];
  }

   // Rest der nicht-leeren Folge:
  if (i==a.length)
    while (j<b.length) c[k++] = b[j++];
  else
    while (i<a.length) c[k++] = a[i++];

  return c;
}
```

```
// *** II.12 Such- und Sortierverfahren mit Arrays

class Hash {

  static int hash (String key, int tableSize) {

    int hashVal = 0;

    for (int i = 0; i < key.length(); i++)
      hashVal = 37 * hashVal + key.charAt(i);

    hashVal %= tableSize;
    if (hashVal < 0)
    hashVal += tableSize;

    return hashVal;
  }

  public static void main (String[] args) {
    String str;
    int length;

    System.out.print("Enter table length: ");
    length = Keyboard.readInt();

    while (true) {
      System.out.print("Enter a string: ");
      str = Keyboard.readString();
      System.out.println("String: " + str
            + "  Hash value: "
            +  hash(str, length));
      if (str.equals("0")) return;
    }
  }
}
```