Klausur
Programmierung

13. 07. 2010
Sommersemester 2010
Studiengang OnlineMedien
Prof. Dr. Bruno Friedmann

Nachname, Vorname _____

Matrikelnummer _____

Studiensemester _____

Studiengang _____

Punkte _____

Note _____

Aufgabe 1 (6+7+7 Punkte)

```
public class NamenArray {
  public static void main(String[] args) {
    String[] strArray = new String[8];

    strArray[0] = "Theobald";
    strArray[1] = "Nicki";
    strArray[2] = "Hanna";
    strArray[3] = "Josefine";
    strArray[strArray.length - 1] = "Ende";

    for (int j = 0; j < strArray.length; j++)
       if (strArray[j] != null)
         System.out.println("A"+j+": "+ strArray[j]);
       else
          System.out.println("A" +j+ ": " + "leer");
     }
}
```

1) Was wird ausgegeben?

2) Schreiben Sie eine Klassen-Methode *namenSuche*. Diese habe als Parameter das zu durchsuchende String Array und den zu suchenden Namen. Der Index des Arrays, indem der Name gefunden wurde, soll zurückgegeben werden. Falls der gesuchte Name nicht im Array steht, soll -1 zurück gegeben werden.

3) Schreiben Sie eine Klassen-Methode *namenSort*. Diese habe als Parameter das zu durchsuchende String Array. Die Namen sollen der Länge nach sortiert werden, d. h. der kürzeste Namen am Anfang, der längste am Ende des Arrays.

   Hinweis 1: die in der Klasse String definierte Methode *length()* gibt die Länge des Strings zurück.

Hinweis 2, möglicher Sortieralgorithmus: Vergleichen Sie beim Array-Durchlauf die Inhalte von 2 benachbarten Feldelementen und tauschen Sie gegebenenfalls. Felddurchlauf sooft wiederholen bis keine Änderungen mehr erfolgen.

Aufgabe 2 (7+7+6 Punkte)

1) Geben Sie die Ausgabe von *main* an! Bitte begründen! Was wird mit *sig* realisiert?

```
public class MatheOp {
  public static void main(String[] args) {
      System.out.println("" + sig(2.4, 1, 3.1415));
      System.out.println("" + sig(2.7, 1, 3.1415));
      System.out.println("" + sig(2.0, 1, 3.1415));
      }

  static double sig(double x, double frq, double ampl)
{
      double amplwert = 0.0;
      double T = 1. / frq;
      double TR = x % T;

      if (TR > 0 && TR <= T / 2)
            amplwert = ampl;
      if (TR > T / 2 && TR <= T)
            amplwert = -ampl;
      return amplwert;
      }
}
```

2) Schreiben Sie eine zusätzliche Methode *sigFeld*, die alle Rückgabewerte von sig für den x-Bereich von 0.0 bis 200.0 mit einer Schrittweite von 0.1 in ein Array schreibt, also für die Werte 0.0, 0.1, 0.2, 0.3, ... 199.7, 199.8, 199.9, 200.0

3) Beschreiben Sie die Wirkungsweise der folgenden Methode!

```
public double funcMax(double a, double b, int nn)
{ // a untere, b obere Grenze des Intervals
 // nn Anzahl der Schritte (n)
    double h = (b-a)/nn;
    double erg = 0;
    erg = func(a);
// func(x) liefert einen Funktionswert zurück
    for (int i = 1; i < nn; i++) {
       erg = Math.max(erg, func(i*h));
    } return erg;
 }
```

Aufgabe 3 (4+4+4+4+4 Punkte)

1) Schreiben Sie eine Objektklasse *TelefonEintrag* mit den Attributen String name und String telNr sowie einen Konstruktor für diese beiden Attribute.

2) Schreiben Sie eine Objektklasse TelefonBuch, mit den folgenden Attributen:

Name	Typ
anzahl	*int*
telbuch	*TelefonEintrag [ ]*

   Das Attribut *anzahl* stehe für die Anzahl der Einträge, das Array *telbuch* dient zur Speicherung der Telefonbuchdaten. Dessen Länge sei 1000. Schreiben Sie einen Konstruktor ohne Parameter, der lediglich die Anzahl auf null setzt.

3) Schreiben Sie zur Klasse *Telefonbuch* eine Objektmethode *hinzufuegen* mit 2 String-Parameter: *Name* und *telNr*. Die Methode soll einen weiteren Eintrag hinzufügen. Ist das Telefonbuch bereits voll (*anzahl* == 1000), so soll die Methode keinen weiteren Eintrag mehr hinzufügen.
   Schreiben Sie eine weitere Methode *hinzufuegen*, jedoch mit einem Parameter vom Typ TelefonEintrag.

4) Schreiben Sie eine Methode *getTelNr* mit einem String-Parameter Name. Diese soll für einen Namen die entsprechende Telefonnummer zurück geben.

5) Schreiben Sie eine Testerklasse, erstellen Sie Objekte von Telefonbuch und TelefonEintrag und Testanweisungen für die Methoden.

Aufgabe 4 (5+5+5+5 Punkte)

1) Schreiben Sie eine abstrakte Klasse *Function*, diese soll die abstrakte Methode *func* enthalten, die als Funktionsmethode für einen bestimmten Wert den resultierenden Funktionswert zurück liefert. *func* hat einen Parameter vom Typ double; der Rückgabewert, der Funktionswert ist ebenfalls double. Stellen Sie sich vor, dass die Methode *funcMax* aus Aufgabe 2 ebenfalls Bestandteil der abstrakten Klasse *Function* ist.

2) Schreiben Sie zur abstrakten Klasse *Function* eine nichtabstrakte Sohnklasse *FuncSic* mit den *public* Attributen *frq* und *ampl* sowie einem Konstruktor. Überschreiben Sie die Methode *func*, indem Sie die Funktion, die mit *sig* in Aufgabe 2 realisiert wird verwenden. Die dort verwendeten Parameter frq und ampl sind nun als Attribute realisiert.
   Hinweis: Sie müssen am Methodenrumpf von *sig* wirklich nichts ändern!

3) Fügen sie zu *Function* die nichtabstrakte Methode *redfunc* hinzu, die, ähnlich *func* einen Funktionswert zurück liefert, der jedoch um einen Faktor reduziert ist. Diese Methode redfunc habe also 2 Parameter, den x-Wert und den Reduktionsfaktor, beide vom Typ double. Benutzen Sie bitte *func*.

4) Schreiben Sie eine Testerklasse für FuncSic.

Aufgabe 5 (6 + 7 + 7 Punkte)

```
import java.io.*;
public class ExcBeispiel {
  public static void main(String[] a) throws IOException {
   int num = einlesen();
   System.out.println("Quadrat von "+num+" ist "+ num*num);
  }

 public static int einlesen() throws IOException {
     BufferedReader stdin = new BufferedReader(
                         new InputStreamReader(System.in));
     String inData = null;
     int zahl = 0;
     boolean goodData = false;

     while (!goodData) {
          System.out.println("Eine Zahl eingeben:");
          inData = stdin.readLine();

          try {
             zahl = Integer.parseInt(inData);
             goodData = true;
          }

          catch (NumberFormatException ex) {
             System.out.println("Ihre Eingabe ist falsch!");
             System.out.println("Bitte nochmals! \n");
          }
       }
       return zahl;
   }
}
```

1) Was geschieht im try &catch Block in Methode *einlesen* des nebenstehenden Programms.

2) Laut API-Doku der Java-API BufferedReader wird die Methode readLine in der Anweisung `inData = stdin.readLine();` eine IOException werfen, „If an I/O error occurs“. Wie wird in nebenstehendem Programm damit umgegangen? Bitte erläutern.

3) Bitte fangen Sie die unter 2) genannte IOException in der Zeile `inData = stdin.readLine();` mit einem try & catch-Block in der main-Methode (!) ab und verändern das Programm, wo nötig!