Prüfung Software-Entwicklung 2

Aufgabensteller: Dr. B. Glavina, Dr. Th. Grauschopf, Dr. S. Hahndel, Dr. U. Schmidt
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Aufgabe 1 (Listen in C; etwa 22%)

a) Deklarieren Sie in C einen Datentyp `struct slist`, welcher ein Listenelement für eine einfach verkettete Liste darstellt, wobei ein Listenelement zwei Gleitkommazahlen doppelter Genauigkeit zur Darstellung von x- und y-Koordinaten, eine Zeichenkette der Länge 20 für die Bezeichnung des Listenelements sowie eine vorzeichenlose ganze Zahl, die Generationsnummer, enthalten soll.

b) Schreiben Sie eine C-Funktion `countInCircle` mit folgendem Prototyp:

```
int countInCircle(struct slist *anker, double abstand);
```

welche die Anzahl aller Punkte in der Liste zurückgibt, die sich in einem Kreis mit dem Radius `abstand` um den Ursprung (0, 0) befinden.

Hinweis: Sie können die Aufgabe sowohl iterativ mit Schleifen, als auch rekursiv lösen.

c) Schreiben Sie eine C-Prozedur `lrotate` mit folgendem Prototyp:

```
void lrotate(struct slist **liste);
```

welche die Liste um eine Stelle nach links rotiert, d.h. das zweite Listenelement wird zum ersten Listenelement und das frühere erste Element wird das neue letzte Listenelement.

Muß für diese Aufgabenstellung wirklich wie oben gezeigt eine Zeigeradresse übergeben werden oder könnte man die Prozedur auch mit dem Prototyp

```
void lrotate(struct slist *anker);
```

realisieren? Begründen Sie Ihre Antwort.

Aufgabe 2 (Klassen in C++; etwa 28%)

Modellieren Sie ein Girokonto als Klasse in C++. Das Konto soll über folgende Eigenschaften verfügen:

- Bei der Einrichtung eines neuen Kontos wird der Name des Inhabers eingetragen, der Kontostand auf 0 gesetzt und ein Standard-Dispositionskreditlimit von 1.000 € eingeräumt.

- Alternativ kann die Kontoeinrichtung mit einer erstmaligen Einzahlung verbunden werden; in diesem Fall kann auch ein vom Standard abweichendes Dispokredit-limit eingeräumt werden, maximal bis zur Höhe der ersten Einzahlung.

- Auf das Konto können jederzeit Einzahlungen geleistet und Abhebungen getätigt werden, letztere allerdings nur bis zum Dispokreditlimit (der das Limit über-steigende Betrag der angeforderten Abhebung wird dann nicht ausgezahlt).

- Kontostand und Dispokreditlimit können abgefragt werden.

- Das Konto wird erst bei Tod des Inhabers gelöscht; ein etwaiges Guthaben verfällt, ein in Anspruch genommener Dispokredit sollte eine Meldung "Konto von <name> um xxx € überzogen" auslösen (der Inhabername ist einzusetzen).

Hinweis: zur Vereinfachung dürfen Sie annehmen, daß alle Geldbeträge in ganzen € angegeben werden – es gibt also keine Cents.

a) Definieren Sie die Klasse `Girokonto`.

b) Implementieren Sie die Methoden von `Girokonto`.

c) Schreiben Sie ein C++-Programm, welches für "Max" und "Moritz" je ein Girokonto anlegt, Einzahlungen auf diese Konten vornimmt, den Kontostand von "Max" abfragt, die Hälfte seines Guthabens (sofern vorhanden) abhebt und auf das Konto von "Moritz" transferiert. Drucken Sie schließlich die Kontostände aus.

Aufgabe 3 (Vererbung und virtuelle Funktionen in C++; etwa 25%)

Verwenden Sie im folgenden sinnvolle Variablennamen, Datentypen und Zugriffs-beschränkungen, soweit nicht vorgegeben. Klassendefinitionen sollten stets einen Konstruktor und - falls sinnvoll - einen Destruktor enthalten, wobei deren Implementierung nicht angegeben werden muss.

Eine Firma entwickelt für den Hausgebrauch ein Programm zur abteilungsbezogenen Kostenkalkulation. Kosten für eine Abteilung sind standardmäßig Personalkosten. Es können weitere Kosten hinzukommen, z.B. Produktionskosten. Für jede Abteilung kann die Kostenkalkulation speziell angepasst werden.

a) Eine Klasse `Abteilung` speichere die Anzahl der Angestellten und eine Reihung dieser Länge mit den jeweiligen Gehältern. Sie hat weiter eine Methode `personalKosten`, die die Summe der Gehälter berechnet, und eine virtuelle Funktion `kosten`, die nur die Personalkosten zurückgibt. Definieren Sie die Klasse `Abteilung`.

b) Implementieren Sie die Methoden `Abteilung::personalKosten` und `Abteilung::kosten`.

c) Die von `Abteilung` abgeleitete Klasse `Produktion` speichert zwei Werte: die Anzahl der erzeugten Produkte und die Stückkosten. Definieren Sie diese Klasse.

d) `Produktion` hat eine eigene Version der Methode `kosten`, die folgenden Wert zurückgibt: Personalkosten + (Anzahl Produkte * Stückkosten).
Geben Sie hierfür eine Implementierung an. Ergänzen Sie ggf. die Klassendefinition aus Teilaufgabe c).

e) Definieren Sie die Klasse `Marketing`. Die Kosten der Marketingabteilung sind nur die Personalkosten. Ist eine eigene Implementierung der `kosten`-Methode nötig, und wie müsste sie ggf. aussehen?

f) In der `main`-Routine werden folgende Variablen deklariert (und anschließend sinnvoll belegt):

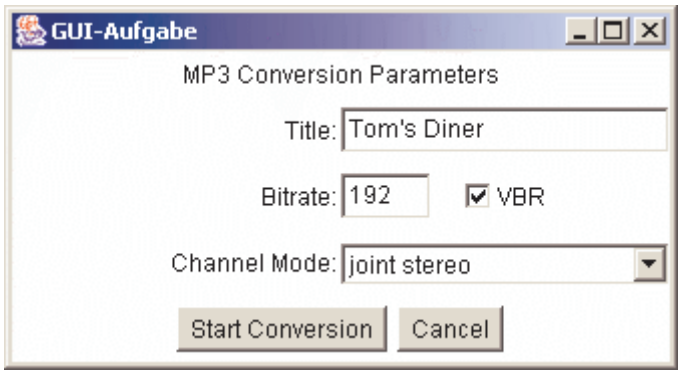
```
Abteilung **abteilungen; // Reihung mit Zeigern
                         // (können auch NULL-Zeiger sein)
int nrAbteilungen;       // Länge der Reihung
```

Schreiben Sie ein Programmstück, das über `abteilungen` iteriert, und die Kosten für die Abteilungen summiert.

Aufgabe 4 (GUI-Programmierung in Java; etwa 25%)

a) Benennen Sie die mit den angegebenen Zeichenketten markierten GUI-Elemente der nachfolgenden Abbildung; geben Sie dabei jeweils die Namen der `awt`-Klasse an. (Hinweis: die Großbuchstaben A bis J dürfen bei Bedarf als Bezug in Ihrer Antwort hier und später verwendet werden.)

```
"MP3 Conversion Parameters" (A)     "Title:" (B)
"Tom's Diner" (C)                   "Bitrate:" (D)
"192" (E)                           "VBR" (F)
"Channel Mode" (G)                  "joint stereo" (H)
"Start Conversion" (I)              "Cancel" (J)
```



b) Skizzieren Sie zur vorstehenden Abbildung die Hierarchie der GUI-Elemente zusammen mit den jeweiligen Layout-Managern.

c) Ergänzen Sie das folgende Programm so, dass das vom Programm erzeugte Fenster der obigen Abbildung gleicht. Sämtliche Ereignisse sollen hier der Einfachheit und Kürze halber ignoriert werden (es kommt NUR auf die graphische Ausgabe an, zusätzlicher Code wird NICHT honoriert!).
Gliedern und kommentieren Sie Ihren Code sinnvoll (das geht in die Punktbewertung ein).

```
/* GUIaufgabe.java */
import java.awt.*;

public class GUIaufgabe
extends Frame
{
  public static void main(String[] args)
  {
    GUIaufgabe myWindow = new GUIaufgabe();
    myWindow.setVisible(true);
  }

  public GUIaufgabe()
  {
    setTitle("GUI-Aufgabe");

    ... // <--- hier wird Ihr Code eingefuegt

  }
} // end class GUIaufgabe

// end file GUIaufgabe.java
```

Hinweis: zur Unterstützung finden Sie im Anschluss an die Prüfungsangabe (als letztes Blatt) einen Ausschnitt aus dem im Praktikum bearbeiteten Programm "GUIdemo.java".

Ausschnitt aus dem Programm "GUIdemo.java" (zu Aufgabe 4c):

```
...
// GUI-Komponente Label:
Panel panel1 = new Panel();
panel1.setLayout(new GridLayout(4,1));
panel1.add(new Label("(siehe Datei \"GUIdemo.txt\"!)"));
panel1.add(new Label("Links",Label.LEFT));
panel1.add(new Label("Zentriert",Label.CENTER));
panel1.add(new Label("Rechts",Label.RIGHT));
// Label sind passive Elemente (koennen keine Ereignisse ausloesen)
//
// GUI-Komponente Button:
Panel panel2 = new Panel();
Button button = new Button("Bing");
button.addActionListener(this); // registriere diese Klasse als ...
   // Ereignisinteressent bei der Ereignisquelle "button"
panel2.add(button); // fuege "button" in "panel2" ein
//
// GUI-Komponente Checkbox:
Panel panel3 = new Panel();
panel3.setLayout(new GridLayout(3,1));
Checkbox cb = new Checkbox("Doppelportion");
cb.addItemListener(this); // register me for events from cb
panel3.add(cb);
cb = new Checkbox("mit Sahne", true);
cb.addItemListener(this);
panel3.add(cb);
cb = new Checkbox("zum Mitnehmen", false);
cb.addItemListener(this);
panel3.add(cb);
//
// GUI-Komponente CheckboxGroup (RadioButtonGroup):
...
//
// GUI-Komponente TextField:
Panel panel5 = new Panel();
TextField tf = new TextField("Meier",20);
tf.addActionListener(this);
tf.addTextListener(this);
panel5.add(tf);
//
// GUI-Komponente TextArea:
Panel panel6 = new Panel();
TextArea ta = new TextArea("Java forever ...", 3, 15);
ta.addTextListener(this);
panel6.add(ta);
//
// GUI-Komponente Choice:
Panel panel7 = new Panel();
Choice choice = new Choice();
choice.addItemListener(this);
choice.add("rot");
choice.add("grün");
panel7.add(choice);
//
// GUI-Komponente List (ListBox):
Panel panel8 = new Panel();
List list = new List(4,false);
list.addActionListener(this);
list.addItemListener(this);
list.add("Äpfel");
list.add("Birnen");
list.select(1);
panel8.add(list);
//
// GUI-Komponente Scrollbar:
...
//
```