Fachhochschule Ingolstadt SS 2004
Studiengang Informatik

Prüfung Software-Entwicklung 2

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

Aufgabe 1 (Zeiger und Geflechte in C; etwa 30%)

Konstruieren Sie einen vereinfachten Familienstammbaum, der für eine gegebene Person nur die direkten Vorfahren enthält, also Eltern, Großeltern, Urgroßeltern usw., aber nicht Geschwister, Onkel, Tanten und andere "indirekte" Verwandten. Jede Person hat einen Vornamen, Nachnamen, Geburtstag und Todestag. Geburtstag und Todestag sind gekennzeichnet durch Angaben für Tag, Monat und Jahr.

a) Entwerfen Sie einen rekursiven Datentyp `person` für Bausteine dieses Stammbaums. Verwenden Sie für die Datumsangaben einen Verbundtyp `datum`. Alle Elemente des Datentyps `person` sollen Zeiger sein.

b) Schreiben Sie eine C-Funktion `ist_urgrossvater(x, y)`, die 1 (`true`) zurückgibt, falls `x` Urgroßvater von `y` ist, andernfalls 0 (`false`). `x` sei genau dann Urgroßvater von `y`, wenn Name und Geburtsjahr von `x` mit den entsprechenden Daten eines der Urgroßväter von `y` übereinstimmen. Sie können bei dieser Teilaufgabe voraussetzen, daß der Stammbaum von `y` bis zur Ebene der Urgroßeltern vollständig besetzt ist.

c) Schreiben Sie eine C-Prozedur `lebende_weibliche_vorfahren(x)`, welche Vornamen, Namen und Geburtsjahr aller noch lebenden weiblichen Vorfahren von `x` auf dem Bildschirm ausgibt.

Aufgabe 2 (Klassen und Vererbung, allgemein und in C++; etwa 25%)

Sie erstellen ein objektorientiertes Modell für die "Mitarbeitertypen" Mitarbeiter, Manager und Vertreter. Diese werden jeweils durch eine gleichnamige Klasse in C++ repräsentiert.

Die Klasse `Mitarbeiter` verfügt über die drei geschützten Attribute `name` (Zeichenkette), `personalnummer` (vorzeichenlose ganze Zahl) und `grundgehalt` (Kommazahl). Zusätzlich steht genau ein Konstruktor zur Verfügung, mit dem es möglich ist, alle drei Attribute der Klasse zu initialisieren, sowie eine öffentliche Methode `get_gehalt()`, die das Gehalt des Mitarbeiters zurückliefert.

Ein Manager unterscheidet sich von einem Mitarbeiter nur dadurch, dass er zusätzlich zum Grundgehalt einen Bonus erhält. Dieser ist als privates Attribut realisiert und darf maximal 30% vom Grundgehalt betragen. Es stehen die beiden öffentlichen Methoden `set_bonus()` und `get_gehalt()` zur Verfügung, wobei letzteres die entsprechende Methode von `Mitarbeiter` überschreiben soll.

Ein Vertreter wird neben seinem Grundgehalt prozentual am Umsatz beteiligt. Dazu verfügt er über zwei zusätzliche private Attribute `provision` und `umsatz` (jeweils Kommazahlen), wobei `provision` nur Werte zwischen 0 und 1 annehmen darf. Es stehen die Methoden `set_umsatz()`, `set_provision()` und `get_gehalt()` zur Verfügung, wobei letztere wieder die gleichnamige Methode von `Mitarbeiter` überschreibt.

Die Klassen `Manager` und `Vertreter` enthalten zusätzlich ebenfalls einen Konstruktor, der alle ihre Attribute vorinitialisiert.

a) Entwerfen Sie ein sinnvolles Klassendiagramm (einschliesslich der Attribute und Methoden) und nutzen Sie auch die Möglichkeiten der Vererbung.

b) Geben sie die Spezifikation aller Klassen in C++ an.

c) Implementieren Sie für alle Klasen die Konstruktoren und die Methode `get_gehalt()`. Implementieren Sie ausserdem für die Klasse `Manager` die Methode `set_bonus()`.

d) (freiwillige Aufgabe, nicht Pflicht) Ist es möglich, in einem Objekt vom Typ `Manager` die Methode `get_gehalt()` der Basisklasse aufzurufen, um an das Grundgehalt ranzukommen. Falls ja, wie geht das? Falls nein, begründen Sie bitte kurz, warum das nicht geht.

Aufgabe 3 (Klassen und Polymorphie in C++; etwa 15%)

Eine Bibliothek will ihren Bestand aus Büchern und Zeitschriften mit einem C++-Programm verwalten.

Eine Klasse `Beschreibung` enthält u.a. beschreibende Daten der Ausleihmedien (Titel, Seitenzahl) und die Methode `getBreite()`:

```
class Beschreibung {
  private:
    char *titel;
    ...
  protected:
    int seitenzahl;
  public:
    ...
    double getBreite();
};
```

Die Klassen `Zeitschrift` und `Buch` sind von `Beschreibung` abgeleitet. Beide überschreiben die Methode `getBreite()`.

a) Welche Zeile welcher Klassendefinition muss ich wie abändern, damit in folgendem Code-Stück die Methode `getBreite()` der Buch-Klasse aufgerufen wird?

```
Beschreibung *p = new Buch(...);
double b = p->getBreite();
```

b) `getBreite()` berechnet in Abhängigkeit von der Seitenzahl die Breite der Zeitschrift oder des Buches. 1000 Buchseiten sind 6 cm breit, 1000 Zeitschriftenseiten sind aufgrund des dünneren Papiers nur 4 cm breit. Geben Sie Implementierungen der Methoden `getBreite()` von `Buch` und `Zeitschrift` an.

Von einem Buch oder einer Zeitschrift kann es mehrere Exemplare geben, z.B. hat die Bibliothek fünf Ausgaben von "Programmieren in C". Deshalb gibt es eine Klasse `Exemplar`, die ein konkretes Exemplar darstellt mit Bestandsnummer und Standort:

```
class Exemplar {
  private:
    char bestandsNummer[8];
    char standort[10];
  public:
    ...
    Beschreibung *getBeschreibung();
};
```

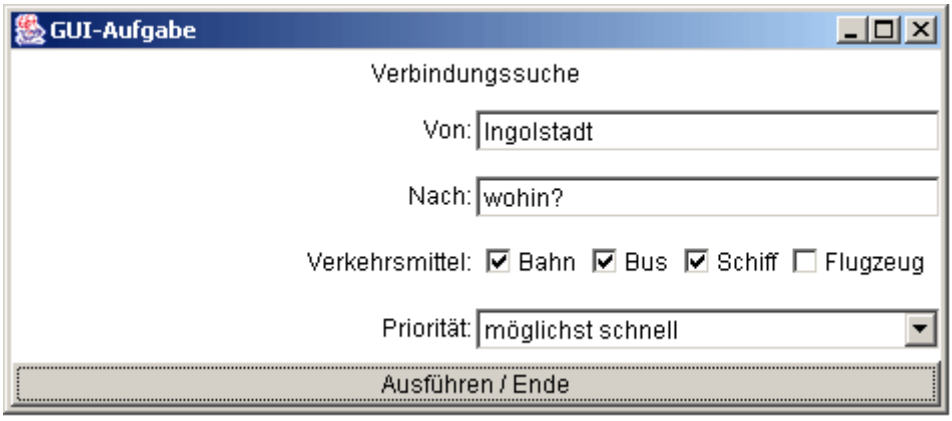
`getBeschreibung()` gibt einen Zeiger (ungleich `NULL`) auf ein Objekt des Typs `Buch` oder `Zeitschrift` zurück.

c) Die Änderung aus Teilaufgabe a kann im Folgenden vorausgesetzt werden. Sie wollen nun die für die Bibliothek benötigten Regalmeter abschätzen. Sie haben eine Reihung `bibliothek` der Länge N, die Zeiger auf alle Exemplare der Bibliothek enthält. Nutzen Sie in einem Code-Stück eine Schleife über diese Reihung, um die Gesamtbreite aller Bücher und Zeitschriften (in Metern) zu berechnen.

Aufgabe 4 (GUI-Programmierung in Java; etwa 30%)

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 M dürfen bei Bedarf als Bezug in Ihrer Antwort hier und später verwendet werden.)

```
"Verbindungssuche" (A)     "Von:" (B)          "Ingolstadt" (C)
"Nach:" (D)                "wohin?" (E)        "Verkehrsmittel" (F)
"Bahn" (G)                 "Bus" (H)           "Schiff" (I)
"Flugzeug" (J)             "Priorität" (K)     "möglichst schnell" (L)
"Ausführen / Ende" (M)
```



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".

Hier folgt der 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:
...
//
```