Fachhochschule Ingolstadt SS 2008
Studiengang Elektro- und Informationstechnik
Studiengang Informatik

Prüfung Praktische Informatik 2
Prüfung Softwareentwicklung 2
Prüfung Grundlagen der Programmierung 2

Aufgabensteller: Prof. Glavina, Prof. Grauschopf, Prof. Hahndel, Prof. Schmidt
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Aufgabe 1: Klassenmodellierung, Vererbung, Polymorphie (ca. 10 %)

Für ein Verwaltungsprogramm der öffentlichen Hand sollen Teile modelliert werden.

a) Stellen Sie die folgenden Zusammenhänge als Klassendiagramm dar (ggf. mit Attributen):
- Eine Person kann eine natürliche oder eine juristische Person sein.
- Eine juristische Person kann ein Unternehmen oder ein Verein sein.
- Vereine können gemeinnützig sein.

b) Stellen Sie ebenfalls als Klassendiagramm dar:
- Ein Immobilienerwerb ist ein Vorgang.
- Eine Geburtenmeldung ist ein Vorgang.

c) Wie modellieren Sie die nun folgenden Anforderungen? Beachten Sie dabei auch die in Teilaufgabe d geforderte Funktionalität.
- Alle Personen (natürlich oder juristisch) außer gemeinnützigen Vereinen sind steuerpflichtig.
- Ein Immobilienerwerb ist steuerpflichtig, eine Geburtenmeldung dagegen nicht.

Zeichnen Sie eventuell nötige zusätzliche Attribute, Methoden, Klassen, ... in die obigen Klassendiagramme ein.

d) Die Verwaltungssoftware soll eine (einzige!) Liste von steuerpflichtigen Personen und Vorgängen halten, um daraus die Steuereinnahmen zu berechnen. Geben Sie eine beispielhafte Definition der Liste und der Implementierung der Steuerberechnung an.

Aufgabe 2: Binärbäume, Programmverständnis (ca. 20 %)

Gegeben sei folgende Definition einer Klasse für Binärbäume.

```
public class BinBaum
{
  private int data;
  private BinBaum left, right;

  public BinBaum(int i, BinBaum l, BinBaum r)
  { data = i; left = l; right = r;
  }

  public BinBaum(int i)
  { this(i, null, null);
  }

  public void printSomeOrder()
  {
    System.out.print("(");
    if (left!=null) left.printSomeOrder();
    System.out.print(data);
    if (right!=null) right.printSomeOrder();
    System.out.print(")");
  }

  public static void main(String[] args)
  {
    BinBaum a, b, c, d;
    System.out.println("start main");
    a = new BinBaum(1);
    b = new BinBaum(2, new BinBaum(3), new BinBaum(4));
    c = new BinBaum(5, new BinBaum(6), null);
    d = new BinBaum(7, b, new BinBaum(8, c, a));
    // Stelle A
    d.printSomeOrder();
  }

} // end class BinBaum
```

a) Skizzieren Sie die Inhalte der Variable a, b, c und d bzw. der Bäume, auf die diese verweisen (im Programmablauf an Stelle A).

b) Geben Sie an, was das Programm ausgibt. Wie nennt man diese Art des Durchlaufens eines Binärbaums?

c) Welche weitere Arten des Durchlaufens von Bäumen kennen Sie? Geben Sie für eine dieser Arten eine Java-Methode zur Ausgabe von Geflechten der Klasse BinBaum an. Was wird ausgegeben, wenn Sie Ihre Methode auf die Variable d anwenden?

Aufgabe 3: Programmierung, Standard-Datenstrukturen (ca. 35 %)

Erstellen Sie für ein Auktionssystem die Klassen `Artikel`, `Benutzer` und `Gebot` wie im folgenden beschrieben. Alle Attribute sollen dabei `private` deklariert werden. Sie dürfen zur Vereinfachung voraussetzen, dass es für jedes Attribut ein Getter-Setter-Paar gibt. Außer für Klasse `Gebot` (Teilaufgabe b) müssen auch keine Konstruktoren definiert werden. (Hinweis: Zur Unterstützung -- bei Bedarf -- finden Sie auf der letzten Seite dieser Prüfungsangabe einen Auszug aus der Java-Dokumentation.)

a) Die Klasse `Gebot` hat als Attribute eine Referenz auf den `Benutzer`, der das Gebot abgegeben hat, die Höhe des Gebots in Cent und den Zeitpunkt, zu dem das Gebot abgegeben wurde. Nutzen Sie für letzteres eine Referenz auf ein `java.util.Date`-Objekt. Geben Sie die Klassendefinition an.

b) `Gebot` hat einen Konstruktor mit zwei Parametern: dem Bieter (Typ `Benutzer`) und der Höhe des Gebots in Cent. Initialisieren Sie damit die entsprechenden Attribute. Das `Date`-Attribut wird durch Aufruf des Default-Konstruktors initialisiert. Geben Sie die Definition des Konstruktors an.

c) `Artikel` hat als Attribute eine Referenz auf den Verkäufer (Typ `Benutzer`), eine Kurzbeschreibung, einen Startpreis in Cent, eine Liste von Geboten und ein Attribut, das anzeigt, ob die Auktion auf diesen Artikel noch läuft, oder bereits beendet ist.
Geben Sie die Klassendefinition an, initialisieren Sie dabei auch die Liste von Geboten.

d) Implementieren Sie für `Artikel` eine Methode `addGebot`, die als Parameter ein `Gebot` erhält. `addGebot` fügt der Liste von Geboten das übergebene `Gebot` am Ende hinzu, wenn

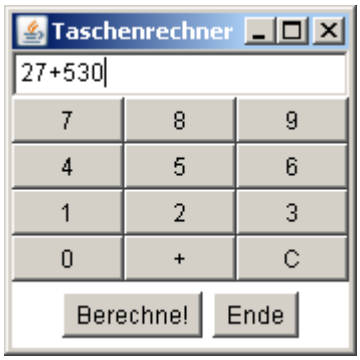
- die Auktion für den Artikel noch läuft,
- das übergebene Gebot bei leerer Gebotsliste höher liegt als der Startpreis, oder
- das übergebene Gebot bei nicht-leerer Gebotsliste höher liegt als das letzte Gebot in der Liste.

Erfolgreiches Hinzufügen des Gebots wird durch Rückgabe eines `boolean`-Wertes signalisiert.

e) Definieren Sie die Klasse `Benutzer` mit Attributen Name, Mail-Adresse und einer Liste namens `verkaufen` von Artikeln, die der Benutzer zur Auktion anbietet bzw. angeboten hat.

f) Implementieren Sie für Benutzer eine Methode `eingenommen`, die zurückgibt, wie viele Cent der Benutzer durch abgeschlossene Auktionen bereits eingenommen hat.

Aufgabe 4: Java-GUI (ca. 35 %)



Es soll eine Oberfläche programmiert werden, die einen (sehr) einfachen Taschenrechner realisiert, der nur addieren kann (siehe Abbildung).

a) Skizzieren Sie die Layout-Hierarchie der GUI-Elemente des gezeigten Fensters. Verwenden Sie dabei eindeutige Bezeichnungen für die Elemente und geben Sie auch alle benötigten Layoutmanager samt Parametern mit an.

b) Schreiben Sie eine Methode

```
private void gb(String btext, Panel p),
```

die einen Button mit der Aufschrift `btext` erzeugt und zum Panel `p` hinzufuegt. Ausserdem soll das aktuelle Objekt als Ereignisempfänger eingetragen werden.

c) Schreiben Sie eine Methode

```
private int calculate(String s),
```

die den String `s` in zwei Teile zerlegt, nämlich in den Teil, der vor dem Pluszeichen steht, und in den Teil, der nach dem Pluszeichen steht. Die beiden Teilstrings werden dann als Zahlen interpretiert und deren Summe wird als Ergebnis zurückgegeben.
(Hinweis: Sie dürfen hierbei davon ausgehen, dass der String String s genau ein Pluszeichen enthält und die Teile vor und nach dem Plus gültige Zahlen darstellen; Fehlerabfragen werden in dieser Aufgabe nicht verlangt. Der Methoden-Aufruf `Integer.parseInt(zs)` liefert den int-Wert des Strings `zs`, sofern `zs` nur aus Ziffern besteht.)

d) Schreiben Sie nun ein Java-Programm, so dass das vom Programm erzeugte Fenster der obigen Abbildung entspricht und das Programm sich bei Bedienung wie folgt verhält:

Drücken von Taste	*Wirkung*
"Berechne!"	Die Anzeige wird als Summe ("x+y") interpretiert und die Summe berechnet und zur Anzeige gebracht.
"Ende"	Das Fenster wird entfernt und das Programm beendet.
"C"	Die Anzeige wird geleert.
Taste ist eines der Zeichen aus "0123456789+"	Das Zeichen wird an den bisherigen Anzeigeninhalt angehängt.

(Hinweis: Verwenden Sie dabei nach Möglichkeit die oben vorbereiteten Hilfsmethoden.)

Auszug aus der Java-Dokumentation (vereinfacht)

```
Klasse java.util.HashMap<K,V>
// K: Typ der Schlüssel (key)
// V: Typ der Werte (value)
```

`HashMap()`	erzeugt leeres Dictionary D (=assoziative Reihung)
`void put(K k, V v)`	trägt das Paar (k, v) in D ein
`V get(K k)`	liefert Wert zu Schlüssel k bzw. null, falls k nicht in D
`Set<K> keySet()`	liefert Menge aller Schlüssel aus D
`void remove(K k)`	entfernt den Eintrag mit Schlüssel k aus D
`void clear()`	leert D (löscht alle Einträge)
`boolean containsKey(K k)`	ist Schlüssel k in D enthalten?
`boolean isEmpty()`	ist D leer?
`int size()`	liefert die Anzahl der Einträge in D

```
Klasse java.util.LinkedList<E>
// E: Typ der Inhalte (element), die die Liste aufnehmen soll
```

`LinkedList()`	erzeugt leere Liste L
`void add(E e)`	hängt das Element e hinten an L an
`void add(int i, E e)`	fügt Element e an Position i ein (Rest verschieben)
`E get(int i)`	liefert Element an Position i
`void set(int i, E e)`	ersetzt das Element an Position i durch das Element e
`void remove(int i)`	entfernt das Element an Position i aus L
`void clear()`	leert L (löscht alle Einträge)
`int indexOf(E e)`	liefert den Index des ersten Auftretens von Element e in L bzw. liefert -1, falls e nicht in L ist
`int lastIndexOf(E e)`	liefert den Index des letzten Auftretens von Element e in L bzw. liefert -1, falls e nicht in L ist
`boolean isEmpty()`	ist L leer?
`int size()`	liefert die Anzahl der Elemente in L