Studiengänge: Informatik, Flug- und Fahrzeuginformatik

Prüfung
Grundlagen der Programmierung 2

Aufgabe	1	2	3	4	5	Summe
Punkte						

Note:

Prüfer: Prof. Dr. B. Glavina, Prof. Dr. Franz Regensburger
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Matrikelnummer:

Studiengang, Semester:

Raum, Platznummer:

Datum:

Dozent:

Dieses Geheft enthält sowohl die Aufgabenstellung als auch den Platz für Ihre Antworten. Schreiben Sie möglichst nur in die vorgegebenen Antwortrahmen.
Geben Sie am Ende der Prüfung wieder alle Blätter ordentlich geheftet ab. Reißen Sie auf keinen Fall einzelne Seiten heraus!

Füllen Sie die erste Seite noch vor der Prüfung mit Ihren persönlichen Angaben aus.

Viel Erfolg!

Aufgabe 1 (Vererbung, etwa 12%)

Gegeben sei ein Programm mit folgenden drei Klassen T, C und S.

```
public class T {
  protected boolean tell = false;
  public T() {
    if (!tell) {
      System.out.println("Tinker");
    } else {
      System.out.println("Dame");
    }
  }
  public String whoAreYou() { return "Tailor"; }
  public String toString()  { return "Koenig"; }
}
```

```
public class C extends T {
  public C() {
    System.out.println(this);
    System.out.println("Soldier");
    tell = true;
  }

  public String toString() {
    System.out.println("von John Le Carre");
    return "Spion";
  }
}
```

```
public class S extends C {
  public S(String who) {
    this();
    if (!tell) {
      System.out.println("alias David John Moore Cornwell");
      System.out.println(who);
    } else {
      System.out.println("alias John Le Carre");
    }
  }

  public S() { System.out.println(this); }

  public String toString() {
    if (!tell) {
      return whoAreYou();
    } else {
      System.out.println("Spy");
      return "by David John Moore Cornwell";
    }
  }

  public static void main(String args[]) {
    new S("Ass");
  }
}
```

Frage: welche Ausgabe erzeugt das Programm bei Ausführung der Methode main() ?

```
// Ausgabe von main() hier angeben:
```

Aufgabe 2 (Comparable, etwa 8%)

Gegeben sei folgende Klasse, die die Weine einer Weinkarte modelliert:

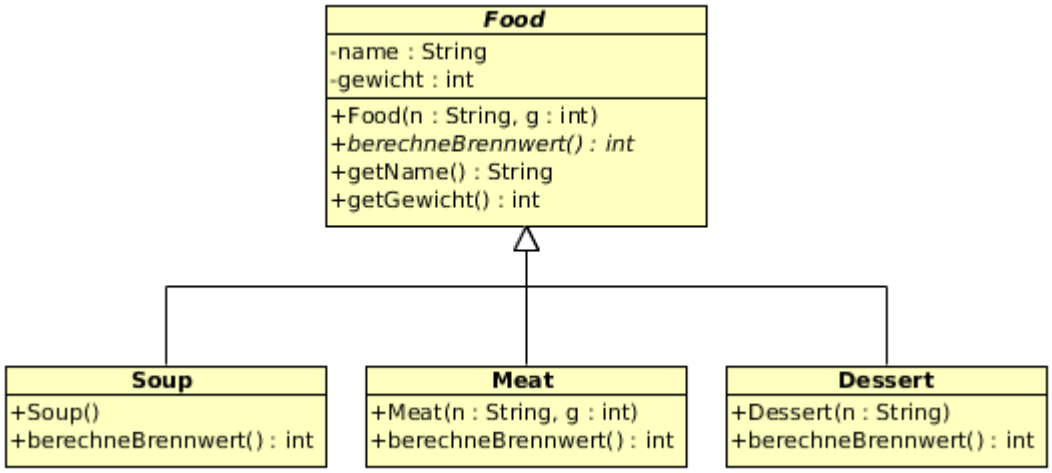
```
public class Wein implements Comparable<Wein> {
  private String name;
  private double preis;

  public Wein(String name, double preis) {
    this.name = name;
    this.preis = preis;
  }
. . .
}
```

Graf Pius von Protz trinkt grundsätzlich immer den teuersten Wein im Restaurant. Sein Stammlokal führt daher eine Weinkarte, auf der die teuersten Weine zuerst aufgelistet werden.
Schreiben Sie für obige Klasse `Wein` die noch fehlende Methode `compareTo()`, die folgende natürliche Ordnung auf den Objekten der Klasse `Wein` definiert:

- teure Weine werden vor billigeren aufgeführt
- bei gleichem Preis sollen die Weine dem Namen nach lexikografisch aufsteigend angeordnet werden

Aufgabe 3 (Polymorphie, etwa 30%)

Gegeben sei folgendes UML-Klassendiagramm:



3a) Schreiben Sie eine Klasse `Food` mit folgenden Vorgaben:

- Privates Attribut `name`, speichert Zeichenreihe beliebiger Länge
- Privates Attribut `gewicht`, speichert Zahl vom Typ `int`
- Konstruktor mit zwei Parametern, die zur Vorbesetzung der Attribute `name` und `gewicht` verwendet werden
- abstrakte Methode `berechneBrennwert()` mit Resultat vom Typ `int`
- zwei Getter `getName()` und `getGewicht()`

3b) Schreiben Sie eine Unterklasse `Soup`, mit

- einem Konstruktor ohne Parameter, der -- unter Verwendung des Oberklassen-konstruktors! -- das Attribut `name` auf `"Soup"` setzt und das Gewicht auf `100`
- einer Implementierung der Methode `berechneBrennwert()`, die konstant den Wert `50` liefert

3c) Schreiben Sie eine Unterklasse `Meat`, mit

- einem zweiparametrigen Konstruktor, der -- unter Verwendung des Oberklassen-konstruktors! -- die Attribute `name` und `gewicht` mit den Parameter-Werten besetzt
- mit einer Implementierung der Methode `berechneBrennwert()`, die das Gewicht mit der Anzahl der Zeichen des Namens multipliziert und das Produkt als Ergebnis liefert

3d) Schreiben Sie eine Unterklasse `Dessert`, mit

- einem einparametrigen Konstruktor, der das Attribut `name` mit dem Parameter-Wert und das Attribut `gewicht` mit dem Wert `200` besetzt -- wieder unter Verwendung des Oblerklassenkonstuktors.
- einer Implementierung der Methode `berechneBrennwert()`, die normalerweise das Gewicht als Ergebnis liefert, es sei denn, dass `"apfel"` oder `"frucht"` im Namen vorkommt, wo sie dann den doppelten Wert des Gewichts als Ergebnis liefert

3e) Die Methode `main()` der Klasse `Poly` sei wie folgt vorgeben:

```
public class Poly {
  public static void main(String[] args) {
    Food[] menue = {
          new Soup(), new Meat("mausragout", 200),
          new Meat("kaenguruhschnitzel", 180),
          new Dessert("pudding"), new Dessert("waldfrucht") };

// hier ergänzen









  }
}
```

Ergänzen Sie den Methodenrumpf von `main()` derart, dass der Brennwert des ganzen Menüs auf der Konsole ausgegeben wird (eine Zahl).

3f) Was würde sich ändern, wenn man alles gleich ließe, aber die Methode `berechneBrennwert()` in der Klasse `Food` nicht mehr abstrakt wäre? Begründen Sie Ihre Antwort!

Aufgabe 4 (Collections, etwa 20%)

Gegeben ist eine endliche Abbildung wie folgt:

```
{ "Adam" ==> 1, "Berta" ==> 1, "Chris" ==> 2, "Dora" ==> 3 }
```

Die Abbildung sei in einem Objekt `notenliste` gespeichert, dessen Typ das Interface `Map<String,Integer>` implementiert.

4a) Ergänzen Sie die endliche Abbildung um drei weitere (sinnvolle!) Einträge. Verwenden Sie hierbei die Notation

```
Name ==> Note
```

um die Abbildung eines Namens auf eine Note auszudrücken.

4b) Können Sie die Umkehrabbildung zur erweiterten Abbildung aus 4a angeben?
Falls ja: Geben Sie diese unten an
Falls nein: Begründen Sie, warum Sie es nicht können.

4c) Geben Sie eine Methode `void printMap(Map<String, Integer> m)` an, die die Notenliste auf der Konsole ausgibt, etwa so:

```
Hugo ==> 2
Egon ==> 5
Eva ==> 1
Hans ==> 1
Kurt ==> 3
Anna ==> 1
```

4d) Geben Sie eine Methode

```
void printNotenstatistik(Map<String, Integer> notenliste)
```

an, die eine Häufigkeitsverteilung der Noten auf der Konsole ausgibt, etwa so:

```
Note  Anzahl
1:    25
2:    2
3:    12
4:    0
5:    3
```

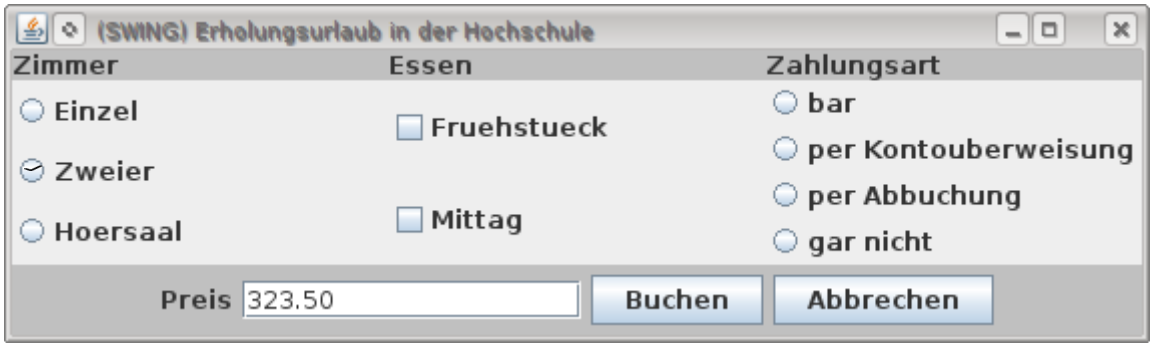
Sie können davon ausgehen, dass nur ganze Zahlen im Bereich `[1,5]` als Noten vorkommen.

Hinweise:
Im Folgenden sind auszugsweise Methoden der Schnittstelle (Interface) Map angegeben. Setzen Sie diese Methoden nach eigenem Ermessen ein.

```
V put(K key, V value)
V get(Object key)
boolean containsKey(Object key)
boolean containsValue(Object value)
Set<K> keySet()
Collection<V> values()
```

Aufgabe 5 (GUI, etwa 30%)

Gegeben sei der folgende Bildschirmabzug:



5a) Überlegen Sie, welche Hierarchie von (sichtbaren und unsichtbaren) GUI-Elementen dieser Oberfläche zugrunde liegt.

Zeichnen Sie (auf der nächsten Seite) die Hierarchie als Baumstruktur und geben Sie zu jedem Baumknoten die Klasse des jeweiligen Elements an; wo nötig, machen Sie auch Angaben zum Layout. Sofern Angaben mehrdeutig sind, achten Sie bitte auf Unterscheidbarkeit (z.B. "Preisschild Hemd" und "Preisschild Hose"). Die funktionale Gruppierung von RadioButtons muss nicht angegeben werden.

Hinweis1:
Sie sollen NUR die Hierarchie zeichnen; eine Programmierung ist hier weder verlangt noch erwünscht.

Hinweis2:
Schauen Sie ZWEIMAL hin, bevor Sie anfangen zu zeichnen:

- warum ist hinter "Einzel" sovie freier Platz und hinter "per Kontoueberweisung" keiner?
- warum stehen "Hoersaal", "Mittag", "gar nicht" in verschiedenen Höhen?

Hier Hierarchie als Baumstruktur einzeichnen. Bei Bedarf Querformat nutzen.

5b) Schreiben Sie ein Java-Programm, das von der oben gezeigten Anwendung NUR die beiden Schaltknöpfe wie folgt realisiert.

Bei Drücken von "Buchen" soll `"Wurde gebucht"` auf die Konsole ausgegeben werden, bei `"Abbrechen"` soll das Programm beendet werden.

Hinweis: Import-Anweisungen müssen nicht angegeben werden