Prof. Dr. Volker Linnemann
Dr. Sven Groppe
Universität zu Lübeck
Institut für Informationssysteme

Lübeck, den 11.1.2008

Vorlesung „Programmieren“

Wintersemester 2007/2008

9. Übungsblatt

1. Mehrfachvererbung (5 Punkte)

Folgende Interfaces seien gegeben, welche Uhren bzw. Radios modellieren sollen:

```
public interface UhrInterface {
      String getTime();
}

public interface RadioInterface {
      int MAX_VOLUME = 10;
      int MIN_VOLUME = 0;
      int getVolume();
      void setVolume(int vol);
}
```

Aufgaben:

a) Schreiben Sie eine Klasse **Radio**, die das Interface **RadioInterface** implementiert. Die momentane Lautstärke soll zu jeden Zeitpunkt mindestens `MIN_VOLUME` und maximal `MAX_VOLUME` (beides inklusive) betragen (1 Punkt)

b) Schreiben Sie eine Klasse **Uhr**, die das Interface **UhrInterface** implementiert. Die Methode `getTime` soll die Systemzeit im Format “`16:05`” zurückgeben.

Hinweis: Die Systemzeit bekommen Sie mit Hilfe der Klasse **java.util.Calendar**. (1 Punkt)

c) Uhrenradios haben sowohl Uhr- als auch Radio-Funktionen. Implementieren Sie eine Klasse **Uhrenradio**, die beide Interfaces **RadioInterface** und **UhrInterface** implementiert. Wie können Sie Ihre Implementierungen (von Uhren und Radios) wiederverwenden? Implementieren Sie zum Testen der Klasse **Uhrenradio** eine Methode, die ein Objekt dieser Klasse erzeugt, die Methoden `getTime`, `setVolume` und `getVolume` aufruft und ggf. das Ergebnis ausgibt. (3 Punkte)

Bemerkungen:

- Dieses ist eine Programmieraufgabe. Bitte geben Sie die Lösung *sowohl* auf **Papier** *als auch* den Java-Quelltext in elektronischer Form per **Email** an Ihren Tutor ab.

2. Abstrakte Datentypen (10 Punkte)

Eine Warteschlange kann eine beliebige Anzahl von Objekten eines Typs `X` entgegennehmen und gibt diese in der Reihenfolge ihres Einfügens wieder zurück („First in, First out“-Prinzip). Der Abstrakte Datentyp `Queue[X]` einer Warteschlange stellt folgende Operationen zur Verfügung:

1: Auf Leere testen:

`empty: Queue[X]` → `Boolean`

2. Warteschlange erzeugen:

`new:` → `Queue[X]`

3. Eintrag ablegen:

`push: X x Queue[X]` → `Queue[X]`

4. Eintrag entnehmen:

`pop: Queue[X]` → `Queue[X]`

5. Kopf auslesen:

`top: Queue[X]` → `X`

(Alternativ kann die Operation `pop` einen Eintrag entnehmen *und* diesen zurückgeben, was wir hier aber nicht weiter betrachten werden.)

Damit ist zum Beispiel

```
top(push(5, push(6, new()))) = 6
```

und

```
empty(pop(push(3, pop(push(5, new()))))) = true.
```

Aufgaben:

a) Entwickeln Sie analog zu den Axiomen zum Abstrakten Datentypen `Stack[x]` (siehe Abschnitt 3.18 der Vorlesung) 4 Axiome für den Abstrakten Datentyp `Queue[X]`. (4 Punkte)

b) Entwickeln Sie eine abstrakte Klasse `Queue`, die die Operationen des Abstrakten Datentyps `Queue[X]` durch Methoden deklariert. Die abstrakte Klasse `Queue` soll dabei die Operationen durch Methoden deklarieren, aber diese noch nicht implementieren! Als Datentyp der Einträge von `Queue` kann `Object` verwendet werden, von dem (auch ohne explizite Angabe der Superklasse) alle Java-Klassen abgeleitet sind. (2 Punkte)

c) Implementieren Sie die abstrakte Klasse in einer konkreten Klasse `WrapperQueue`, die die Klasse `java.util.LinkedList` verwendet, um die Funktionalitäten der Operationen zur Verfügung zu stellen. (4 Punkte)

Bemerkungen:

- Dieses ist eine Programmieraufgabe (Teilaufgaben b) und c)). Bitte geben Sie die Lösung für die Teilaufgaben b) bis c) *sowohl* auf **Papier** *als auch* den Java-Quelltext in elektronischer Form per **Email** an Ihren Tutor ab.
- Die Klasse `java.util.LinkedList` implementiert das Interface `java.util.Queue`, welches ebenfalls Operationen für eine Warteschlange in (zu dieser Aufgabe verschiedenen) Methoden deklariert. In Teilaufgabe c) entwickeln Sie demnach eine Klasse, die nach außen die Methoden der Klasse `Queue` zur Verfügung stellt, aber nach innen die in `java.util.Queue` deklarierten Methoden verwendet. Solche Art von Klassen werden auch Wrapper-Klassen genannt und werden oftmals bei der Integration von heterogenen Softwareprojekten verwendet.

Bemerkungen:

- Jede Seite soll oben rechts den Namen der Abgebenden und die Übungsgruppennummer (wichtig!) enthalten.
- Lösungen für die Übungsaufgaben sind (in der Regel) zu zweit abzugeben.

- Kommentieren Sie Ihre Lösungen! Besteht eine Lösung aus mehreren Zetteln, so sind diese zusammen zu heften. Bitte keine Hüllen, Mappen, o.ä..
- Bitte schicken Sie *Programmieraufgaben zusätzlich zur Abgabe auf Papier in elektronischer Form per Email* an Ihren jeweiligen Tutor.
- Kommentieren Sie ihren Quelltext bei Programmieraufgaben. Dabei sollen keine Trivialitäten kommentiert werden, also bitte keine Kommentare wie

  ```
  x=5; // Wir weisen nun der Variablen x den Wert 5 zu
  ```

  sondern sinnvolle Kommentare, die Ideen des Quelltextabschnittes beschreiben oder auf Unteraufgaben (z. B. a), b), …) hinweisen.
- **Hinreichende Bedingung für die Zulassung zur Klausur:** 50% der erreichbaren Punkte bei jedem Übungszettel (bis auf zwei) und einmaliges Vorrechnen in der Übung
- **Zertifikatskriterium:** Das Bestehen der Klausur am Ende des Semesters

Abgabetermin: Freitag, 18.1.2008, nach der Vorlesung