Übungen zur Vorlesung Programmieren 1
FHWS – Fakultät für Informatik und Wirtschaftsinformatik

Übungsblatt 11

Aufgabe 1: (Ausweis)

a) Schreiben Sie die Klasse Ausweis mit den Attributen Vorname, Nachname und Geschlecht. Neben einem parameterbehaftetem Konstruktor soll die Kasse die entsprechenden getter- und setter-Methoden besitzen. Weiterhin soll die Klasse die Methode toString() überschreiben, welche die Inhalte der Instanzattribute auf der Standardausgabe anzeigt.

Erzeugen Sie zwei weitere Klassen Studentenausweis und Vereinsausweis welche von der Klasse Ausweis abgeleitet sind, und weitere klassenspezifische Attribute besitzen. Weiterhin soll die geerbte Methode toString() in den Subklassen geeignet überschrieben werden.

b) Schreiben Sie eine Methode namens erzeugeAusweise, welche ein Feld erzeugt in dem gleichzeitig bis zu 6 Instanzen von der Klasse Ausweis, Studentenausweis oder Vereinsausweis abgelegt werden können. Das Feld soll 2 Instanzen der Klasse Ausweis, 2 Instanzen der Klasse Studentenausweis und 2 Objekte der Klasse Vereinsausweis beinhalten. Der Rückgabewert der Methode ist das innerhalb der Methode erzeugte Feld.

c) Schreiben Sie eine private Methode namens datenAusgabe, welcher als Parameter ein Feld unbekannter Länge übergeben wird, in dem Objekte der Klasse Ausweis, Studentenausweis oder Vereinsausweis abgelegt sind. Im Rumpf der Methode sollen die Daten aller im Feld enthaltenen Objekte ausgegeben werden. Beachten Sie, dass nicht alle Feldelemente im übergebenen Feld zwingend belegt sein müssen. Für den Fall, dass Feldelemente nicht belegt sind, ist eine entsprechende Meldung in der Form: „Dieser Index <index> ist nicht belegt“ mit Angabe des Feldindex auszugeben.

Aufgabe 2: (Uhrzeit)

a) Schreiben Sie eine Klasse namens Uhr, die die Uhrzeit ((24-)Stunden, Minuten, Sekunden) verwaltet und folgende Methoden enthält:

- *Uhr (int std, int min, int sek)* // einziger Konstruktor

Setzt die Uhrzeit auf die angegebenen Paramter. Falls mind. einer der Parameter ungültig ist, soll die Uhrzeit auf genau 12:00:00 (= $12^{\underline{00}}$ Uhr und 0 Sekunden) gesetzt werden.

- *int getStd ()*

Liefert die Stunden der Uhrzeit zurück

- *int getMin ()*

Liefert die Minuten der Uhrzeit zurück

- *int getSek ()*

Liefert die Sekunden der Uhrzeit zurück

- *void setUhr (int std, int min, int sek)*

Setzt die Uhrzeit auf die angegebenen Parameter. Falls mind. einer der Paramter ungültig ist, soll eine Fehlermeldung ausgegeben werden und die original Uhrzeit unverändert bleiben.

- *void naechsteSek ()*

Zählt die Uhrzeit um 1 Sekunde hoch.

- *long getTagesSek ()*

Liefert die Uhrzeit (seit Mitternacht) komplett in Sekunden zurück.

- *boolean wecker (Uhr weckzeit)*

Überprüft, ob die Uhrzeit mit der Zeit von *weckzeit* übereinstimmt.

Hinweis: 1 Minute hat 60 Sekunden. 1 Stunde hat 60 Minuten.

b) Vereinbaren Sie ein Feld *weltzeit* für 24 verschiedene Uhren. Erzeugen Sie anschließend als eigenständige Anweisung(en) 24 verschiedene Uhren für dieses Feld und initialisieren Sie die erste Uhr im Feld mit der Uhrzeit 0:23:07 (= 0^{23} Uhr und 7 Sekunden), die zweite Uhr mit 1:23:07, usw. bis 23:23:07.

c) Wie sieht die Schleife aus, um im Feld *weltzeit* aus Teilaufgabe b) jede der 24 Uhrzeiten um 1 Sekunde hoch zu zählen?

Aufgabe 3: (Bahn-Modellierung)

Bei der Deutschen Bahn fahren verschiedene Zugtypen. Alle Zugtypen sollen über eine Lok und eine bestimmte Anzahl Wagons verfügen. Ein Wagon für Güter hat i.d.R. ein Fassungsvermögen und transportiert eine bestimmte Art an Gütern, z.B. Autos, Gefahrengut, o.ä.

In einem Personenzug gibt es verschiedene Arten von Wagons: ein Bordbistro, ein Wagon für Passagiere und den Fahrradwagen. Hier hat jeder Wagon eine Anzahl Sitzplätze. In den ICEs und ICs können zusätzlich sogar Sitzplatzreservierungen durchgeführt werden.

a) Entscheiden Sie, welche Klassen inkl. Attribute Sie benötigen, um dieses Problem in einem Java Programm abzubilden.

b) Zeichnen Sie ein Diagramm, in welchem die Beziehungen (Vererbung, Komposition, Aggregation, Assoziation) zwischen den Klassen inkl. ihrer Attribute gezeigt werden.

Aufgabe 4: (Bahn-Implementierung)

Schreiben Sie ein Java Programm, mit dem ein Benutzer selbst einen Zug (modelliert in Aufgabe 3) zusammensetzen kann. D.h. der Benutzer soll in einem Menü wiederholt auswählen, welchen Wagon er als nächstes an den Zug anhängen möchte, z. B.:

1: Lok hinzufügen
2: Passagierwagon hinzufügen
3: Fahrradwagen hinzufügen
4: Güterwagen hinzufügen
5: Bordbistro hinzufügen
6: fertig

Hinweis: Sie haben sowohl bei der Modellierung (Aufgabe 3) als auch bei der Implementierung (Aufgabe 4) einen größeren Freiheitsgrad!