Übungszettel 3. Termin, 28.10.2006 und 4. Termin, 11.11.2006

Eine Mini-CD-Verwaltung

Ein Musikfan führt Buch über den Verleih seiner CDs an seine Freunde. Erstellen Sie eine Java-Applikation für einen Musikfan. Dieser Fan besitzt viele aktuelle CDs und ist deshalb sehr beliebt. Um den Überblick nicht zu verlieren, will er in seinem PC über seine CDs und die Ausleihen Buch führen. Über die grafische Benutzungsoberfläche des Programms können zwei Aktionen angestoßen werden: Ausleihe einer CD und Rückgabe einer CD.

Schauen Sie sich die folgende kleine unvollständige CD-Verwaltung an. Anhand der Kommentare und Ihrem bisherigem Wissen müssten Sie inzwischen in der Lage sein die Funktionsweise des Programms zu verstehen:
CDAnwendung.java, CD.java, CDInputDialog.java, CDTableModel.java

Aufgabe 0: Speichern Sie alle Klassen in ein Verzeichnis und kompilieren Sie das Programm und führen es aus. Sehen Sie sich den Quellcode aller Klassen an und versuchen Sie die Funktionsweise des Programms nachzuvollziehen.

Wie sie sehen besteht die Anwendung bis jetzt aus den folgend aufgelisteten Bestandteilen, die schon einen Teil der gesamten Aufgaben des vollständigen Programms erledigen können:

1) CDAnwendung.java: In der Anwendung werden die Menüs definiert und eine Tabelle ins Fenster gezeichnet.
2) CD.java: Die Fachkonzeptklasse CD mit nur einem Attribut.
3) CDInputDialog.java: Das Dialogfenster für die Eingabe der CD.
4) CDTableModel.java: Die Definition der Eigenschaften und Fähigkeiten der Tabelle.

Dieser Anwendung fehlen zunächst folgende Möglichkeiten:
A: Der Beenden-Menue-Punkt funktioniert nicht.
B: Der Abbrechen-Knopf des Dialogs funktioniert nicht.
Wenn Sie diese Anwendung weiter entwickeln wollen, gehen Sie am besten Schrittweise mit kleinen Schritten vor. Halten Sie Ihr Programm nach jeder Änderung funktionsfähig !

Aufgabe 1: Fügen Sie dem Menüpunkt "Beenden" eine Aktion hinzu, die das Programm mit Hilfe des Aufrufs System.exit(0); beendet.

Aufgabe 2: Mit der Methode dispose() (beseitigen) lässt sich ein Fenster löschen um mit dem Abbrechen-Knopf den Dialog zu schließen. Benutzen Sie die Methode dispose() um den Abbrechen-Knopf des Eingabe-Fensters mit einer Aktion zu verbinden, die das Eingabe-Fenster beseitigt.

Die Tabelle

Schauen Sie sich nun die Klasse CDTableModel.java an. Hier werden die Eigenschaften der Tabelle festgelegt. In dieser Klasse wird z.B. definiert, wie die Spalten-Überschriften der Tabelle lauten und wie die Tabellen-Elemente ermittelt werden.

Aufgabe 3: Ändern Sie die Anzahl der Zeilen und Spalten der Tabelle. Ändern Sie die Überschriften und Inhalte der Tabelle.

Viele CDs

Jetzt wollen wir statt wie bisher nur eine CD eine beliebige Anzahl an CDs verwalten. Schauen Sie Sich dazu zunächst die Container-Klasse an, die mit Hilfe der Klasse Vector die CD-Sammlung beinhaltet: CDContainer.java

Durch den folgenden Aufruf können Sie dem Container eine neue CD hinzufügen:

```
CD newAlbum=new CD();
newAlbum.setTitel("Mensch");

CDContainer.getInstance().addCD(newAlbum);
```

In den ersten beiden Zeilen wird eine CD angelegt und das Attribut geändert. Danach wird die CD mit **CDContainer.getInstance().addCD(newAlbum);** in die Sammelung aufgenommen.

Aufgabe 4: Das Eingabe-Fenster ändert die Attribute einer neu angelegten CD. Fügen Sie diese CD dem Container hinzu. (Hinweis: Die Variable der neu angelegten CD heißt in der Mini-Verwaltung nicht newAlbum.)

Für Testzwecke ist in der Klasse CDContainer.java die Methode **TestAusgabe()** programmiert, welche alle Titel der CDs in der Sammlung auf dem Bildschirm ausgibt. Damit sie sehen, dass die CDs der Sammlung hinzugefügt wurde, lassen Sie sich an der Stelle wo Sie die CD dem Container hinzufügen die Sammlung mit dem folgenden Befehl ausgeben:

```
CDContainer.getInstance().TestAusgabe();
```

Aufgabe 5: Fügen Sie ihrem Eingabe-Fenster-Objekt eine Möglichkeit hinzu, die es erlaubt abzufragen, ob der OK-Button gedrückt worden ist (oder der Abbrechen-Button). In Abhängigkeit davon können Sie dann in der Anwendung die CD hinzufügen oder nicht.

Das ist am einfachsten in der Klasse **CDInputDialog.java** durch hinzufügen des Attributs **okstatus** und der Methode **getOKstatus()**, die den Status zurückliefert, zu erledigen:

```
boolean okstatus = true;

 boolean getOKstatus()
        {
        return okstatus;
        }
```

Frage: An welcher Stelle sollte man das Attribut okstatus auf false setzen?

Die Tabelle mit dem Container verbinden

Um die Tabelle mit dem Container zu verbinden, fangen wir damit an, die Tabellenzeilen mit der Anzahl der CDs in dem Container zu verknüpfen. Die Anzahl der CDs erhalten Sie über den CD-Container mit:

```
CDContainer.getInstance().getCDNumber();
```

Aufgabe 6: Ändern Sie das Tabellenmodell so, dass die Zeilenzahl der Tabelle dem Wert der bisher gespeicherten CDs entspricht. (Sie sehen, dass die Tabelle aber leider nicht länger wird, wenn man eine CD hinzugefügt hat.)

Damit die Tabelle auch aktualisiert wird, muss man der Tabelle mitteilen, dass sich etwas an ihr geändert hat. Dazu dient die Klasse **TableModelEvent**, die Sie im Folgenden speziell an der MiniCD-Verwaltung kennen lernen sollen:

Zunächst muss die Klasse **TableModelEvent** in der Anwendung (CDAnwendung.java) importiert werden:

```
import javax.swing.event.TableModelEvent;
```

Der Anwendung (CDAnwendung.java) sollte ein weiteres Attribut hinzugefügt werden:

```
TableModelEvent mytme;
```

Wenn Ihre Anwendung startet (hier in jbInit()) müssen Sie das Tabellen-Modell mit dem Tabellen-Ereignis verknüpfen, damit man der Tabelle eine Änderung mitteilen kann. Das geht hier mit der Befehlszeile:

```
mytme = new TableModelEvent(unserTabellenModell);
```

An der Stelle, wo sich die die Tabelle ändern soll, können Sie der Tabelle mitteilen, dass sich die Tabelle geändert hat. Sie zeichnet sich darauf hin neu. Das wird in unserem Beispiel mit dem folgenden Befehl gemacht:

```
CDTabelle.tableChanged(mytme);
```

Aufgabe 7: Fügen Sie die das Attribut **TableModelEvent mytme;** der Anwendung hinzu und teilen Sie Ihrer Tabelle mit (**CDTabelle.tableChanged(mytme);**), dass sie sich neu zeichnen soll, wenn dem Container eine CD hinzugefügt wurde.

Aufgabe 8: Ändern Sie Ihr Tabellenmodell so, dass in der Tabelle die Titel der CDs erscheinen.

Kleine Hilfe: Zunächst muss in der Methode **getValueAt(int row, int col)** die CD an der Stelle row aus dem Container geholt werden:

```
CD theCD = CDContainer.getInstance().getCD(row);
```

Dann können Sie die den Titel der CD bestimmen und zurückliefern:

```
return  theCD.getTitel();
```

Speichern der CDs

Jetzt wollen wir, dass die CDs nicht beim Schließen des Programms verloren gehen. Dazu sind schon ein paar Vorbereitungen getroffen worden. Die Klasse CD ist serialisierbar gemacht worden (Serializable). Dadurch ist es möglich die gesamte Sammlung (alle CDs in dem Vektor der Container-Klasse) mit einem mal abzuspeichern.
Die folgenden Methoden **speichern()** und **laden()** können Sie direkt in Ihrem Programm benutzen, um die Klasse CDContainer um die Möglichkeit des Speicherns und Ladens zu erweitern. (Für Sie ist hier wichtig, dass der Dateiname angegeben werden kann und Sie den gesamten Vektor mit einem mal abspeichern können.)
Um die Ein-Ausgabe-Methoden benutzen zu können, müssen die Ein-Ausgabe-Klassen importiert werden.

```
import java.io.*;
void speichern(String filename)
   {
   try {
     //Filenamen setzen
     FileOutputStream fs = new FileOutputStream(filename);
     ObjectOutputStream os = new ObjectOutputStream(fs);
     // Das Objekt lnkCD (den Vektor) auf die Platte schreiben
     os.writeObject(lnkCD);
     os.close();
      }
   catch (IOException exc) {
         System.err.println(exc.toString());
      }
   }
 void laden(String filename)
    {
    try {
        // Filnamen festlegen
        FileInputStream fs = new FileInputStream(filename);
        ObjectInputStream is = new ObjectInputStream(fs);
        // Das eingelesene Objekt ist der Vector, der alle
gespeicherten CDs enthaelt.
        lnkCD=(Vector)is.readObject();
        is.close();
        }
   catch (ClassNotFoundException e) {
         System.err.println(e.toString());
         }
   catch (IOException exc) {
        System.err.println(exc.toString());
        }
    }
```

Nebenbemerkung: Die so genannten try-catch-Bloecke dienen der Behandlung von evtl. auftretenden Fehlern bei der Ein- oder Ausgabe.

Aufgabe 9: Erweitern Sie die Klasse CDContainer.java um die Methoden speichern() und laden().

Die neu eingefügten Methoden lassen sich wie folgt in der Anwendung (CDAnwendung.java) benutzen:

Speichern aller CDs in die Datei "daten.ser" , die in (dem Vektor) der Container-Klasse vorhanden sind:

```
CDContainer.getInstance().speichern("daten.ser");
```

Laden der CDs, die in der Datei "daten.ser" gespeichert sind:

```
CDContainer.getInstance().laden("daten.ser");
```

Aufgabe 10: Fügen Sie der Mini-Verwaltung die Menü-Punkte Öffnen und Speichern hinzu. Verknüpfen Sie die Menüpunkte mit den Methoden laden und speichern. Fangen Sie zunächst mit dem Speichern an.