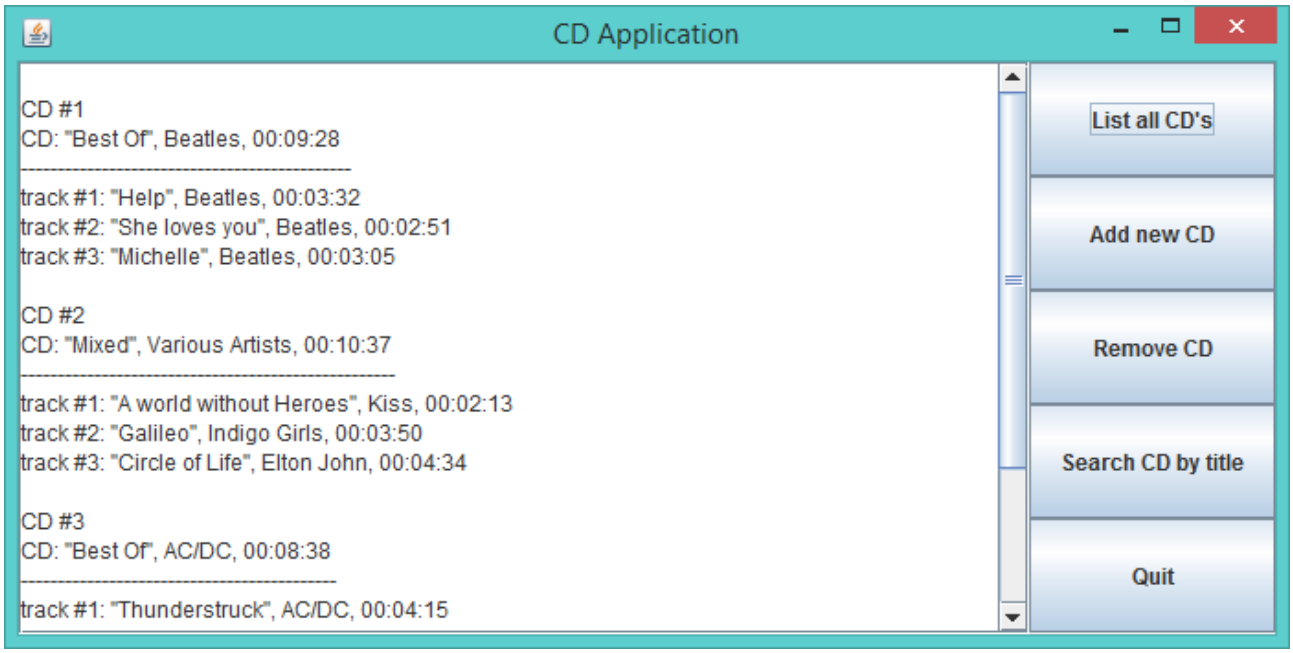
Exercise 10.02

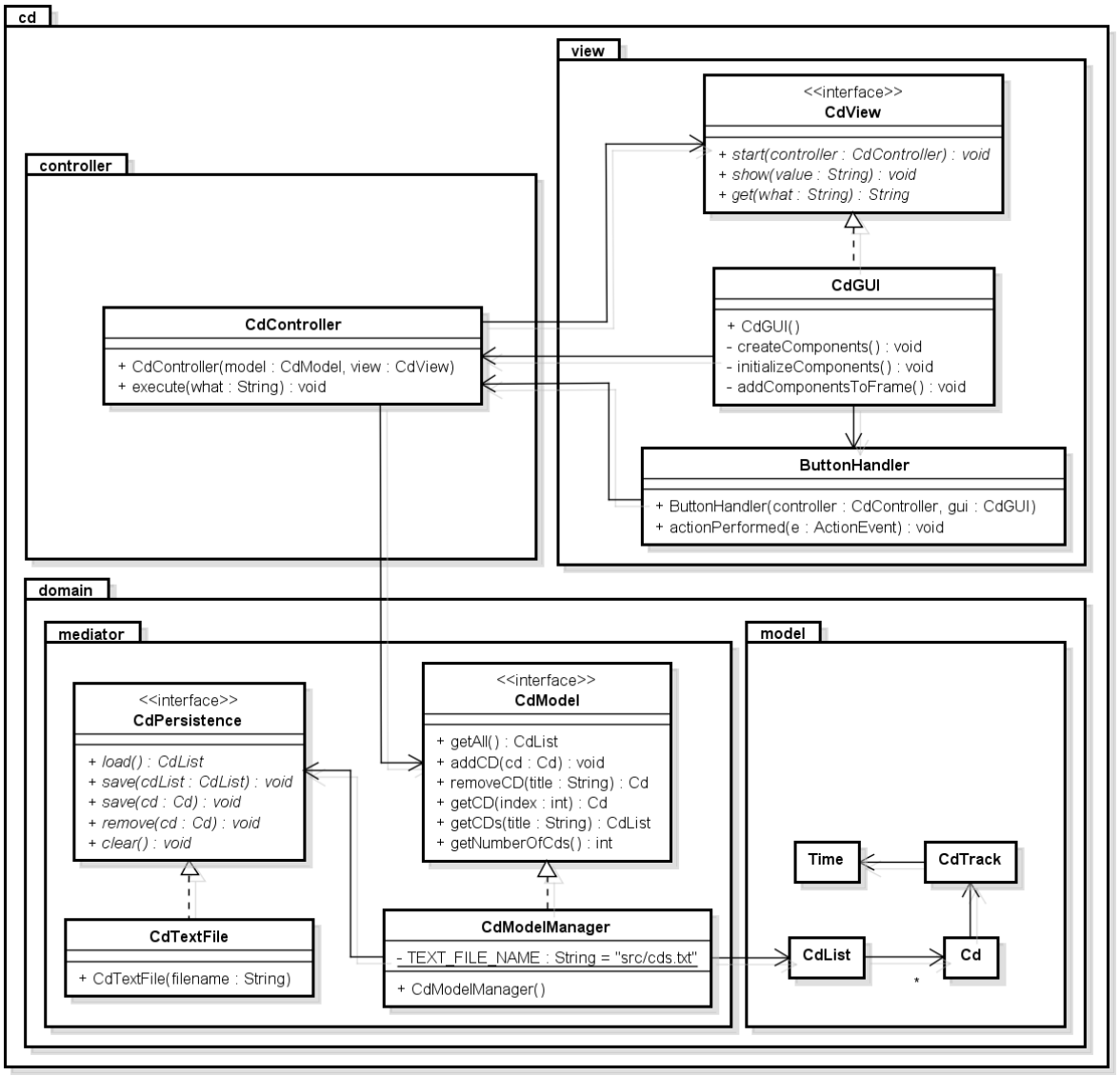
Download `CD-GUI.zip` including an Eclipse project for the CD application from the slides. Import this into Eclipse. Run the class `Main`.

After running class Main and clicking the "List All CDs" button the output should be like the following:



Exercise 10.03

The Exercise is to convert an already existing application (given in `CD-GUI.zip`) to a Model-View-Controller version exactly as shown in the class diagram below



You have to do the following:

- Packages
  - Move `CdList`, `Cd`, `CdTrack` and `Time` into package `cd.domain.model`
  - `CdPersistence` and `CDTextFile` in class `cd.domain.mediator`
  - `CdView` and `CdGUI` in package `cd.view`
- Create interface `CdModel` (in `cd.domain.meditor` package) with the following methods:

```
public CdList getAll();
public void addCD(Cd cd);
public Cd removeCD(String title);
public Cd getCD(int index);
public CdList getCDs(String title);
public int getNumberOfCDs();
```

- Create class `CdModelManager` (in `cd.domain.meditor` package) The Model Manager keeps the Model's state – in this case only a `CdList` instance variable. The second instance variable is a `CdPersistence`.

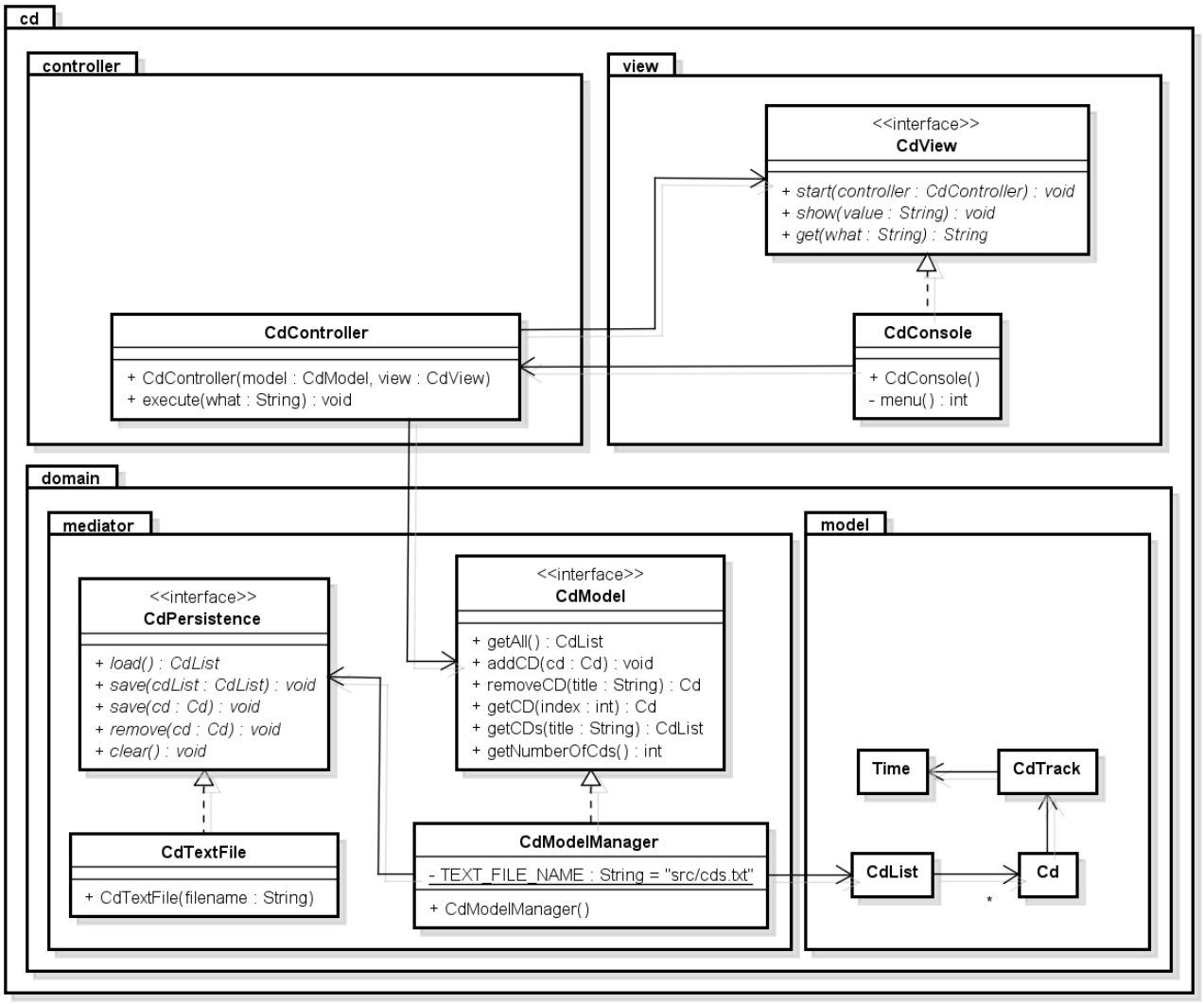
    - The constructor is loading the model/CdList from file (and this part is then deleted from class `Main`)
    - All methods are simply delegating the work to `CdList` and could each be implemented with a single statement.
- Class `CdController` (in package `cd.controller`) has
    - A constructor taking the model and the view
    - A method with the logic from the `ButtonHandler`'s method `actionPerformed`.
- Class `CdGUI` has no longer a reference to the model (to class `CdList)` and method `start()` is changed to include a `CdController` as argument.
- Class `ButtonHandler` has now a `CdContoller` as instance variable and `actionPerformed` calls method `execute(…)` in class `CdController` (and do not include any actions).
- Change `Main` to the following

```
import cd.domain.mediator.*;
import cd.view.*;
import cd.controller.*;

public class Main
{
   public static void main(String args[])
   {
      CdModel model = new CdModelManager();
      CdView view = new CdGUI();
      CdController controller = new CdController(model, view);
      view.start(controller);
   }
}
```

(Exercise 10.04)

The Exercise is to convert an already existing application (the application from `CD-Console.zip`) to a Model-View-Controller version exactly as shown in the class diagram below



Use the previous exercise as basis because this will give you less work.

A few notes:

- The model is unchanged (Model package with all classes and class `CdModelManager`). The same goes for interfaces `CdModel`, `CdPersistence`, `CdView` and class `CdTextFile`.
- Class `CdController`
    - Method `execute(…)` takes a String as argument. This is either `"List"`, `"Add"`, `"Remove"`, `"Search"` or `"Quit"`. Use e.g. a switch to act upon this input. The logic is the same as in the `CdConsole` method `start(…)`.
- Change `Main` to the following

```
import cd.controller.CdController;
import cd.domain.mediator.CdModel;
import cd.domain.mediator.CdModelManager;
import cd.view.CdView;
import cd.view.CdConsole;

public class Main
{
  public static void main(String args[])
```

```
  {
    try
    {
      CdModel model = new CdModelManager();
      CdView view = new CdConsole();
      CdController controller =new CdController(model, view);

      view.start(controller);
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
  }
}
```