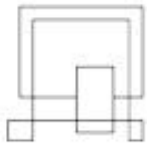
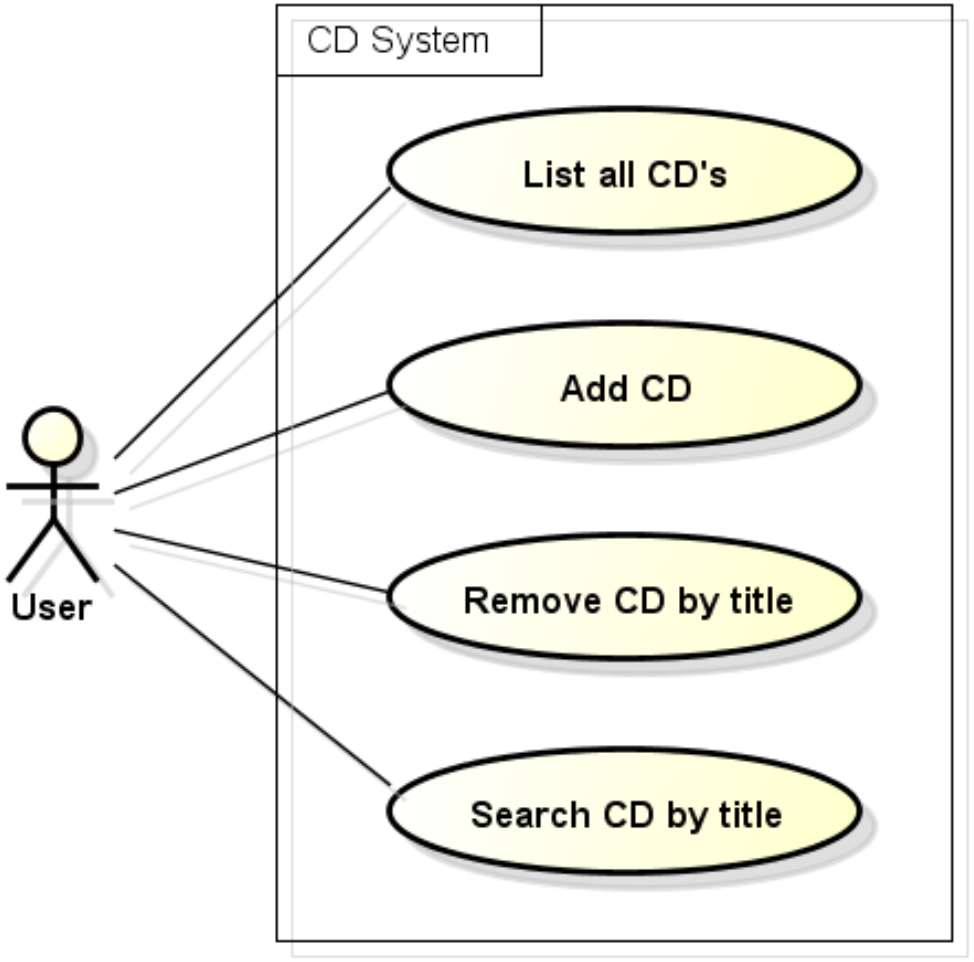
Life is great
VIA University College



Software Development with UML and Java 2

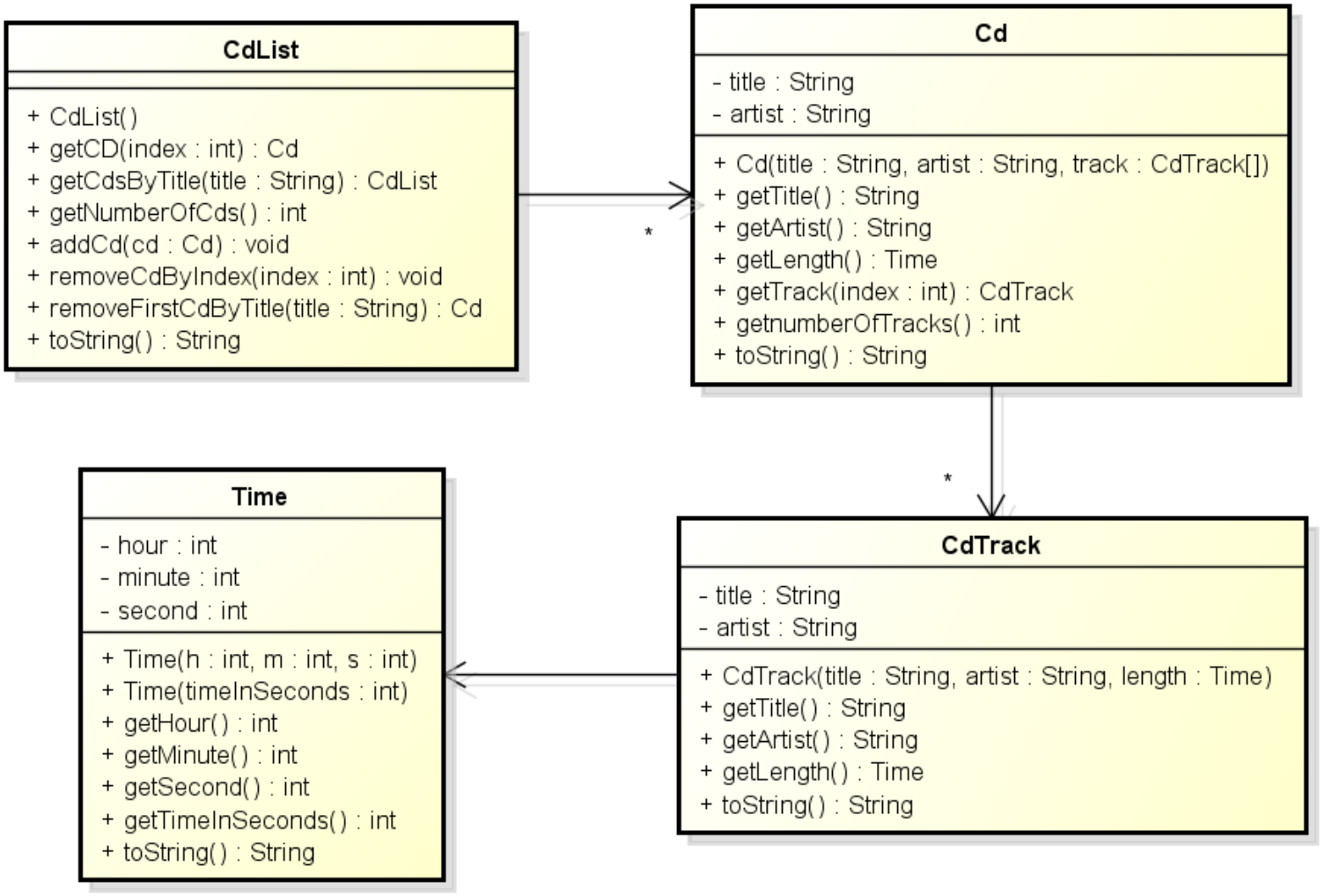
Example (of a tiny system)



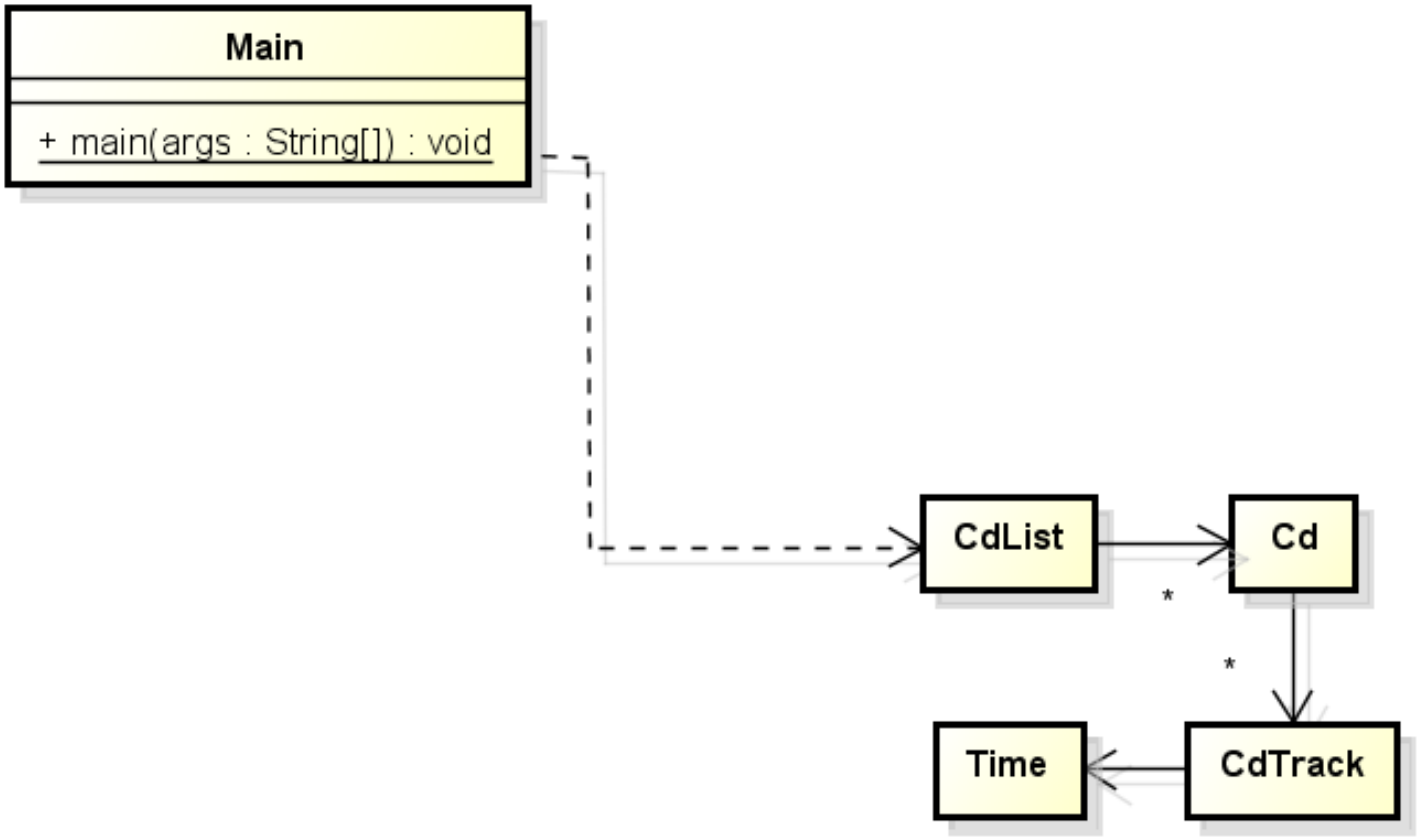
Example: CD Application (Console)

```
CD Application
--------------
1) List all CD's
2) Add new CD
3) Remove CD by title
4) Search CD by title
5) Quit

Select an item 1-5:
```

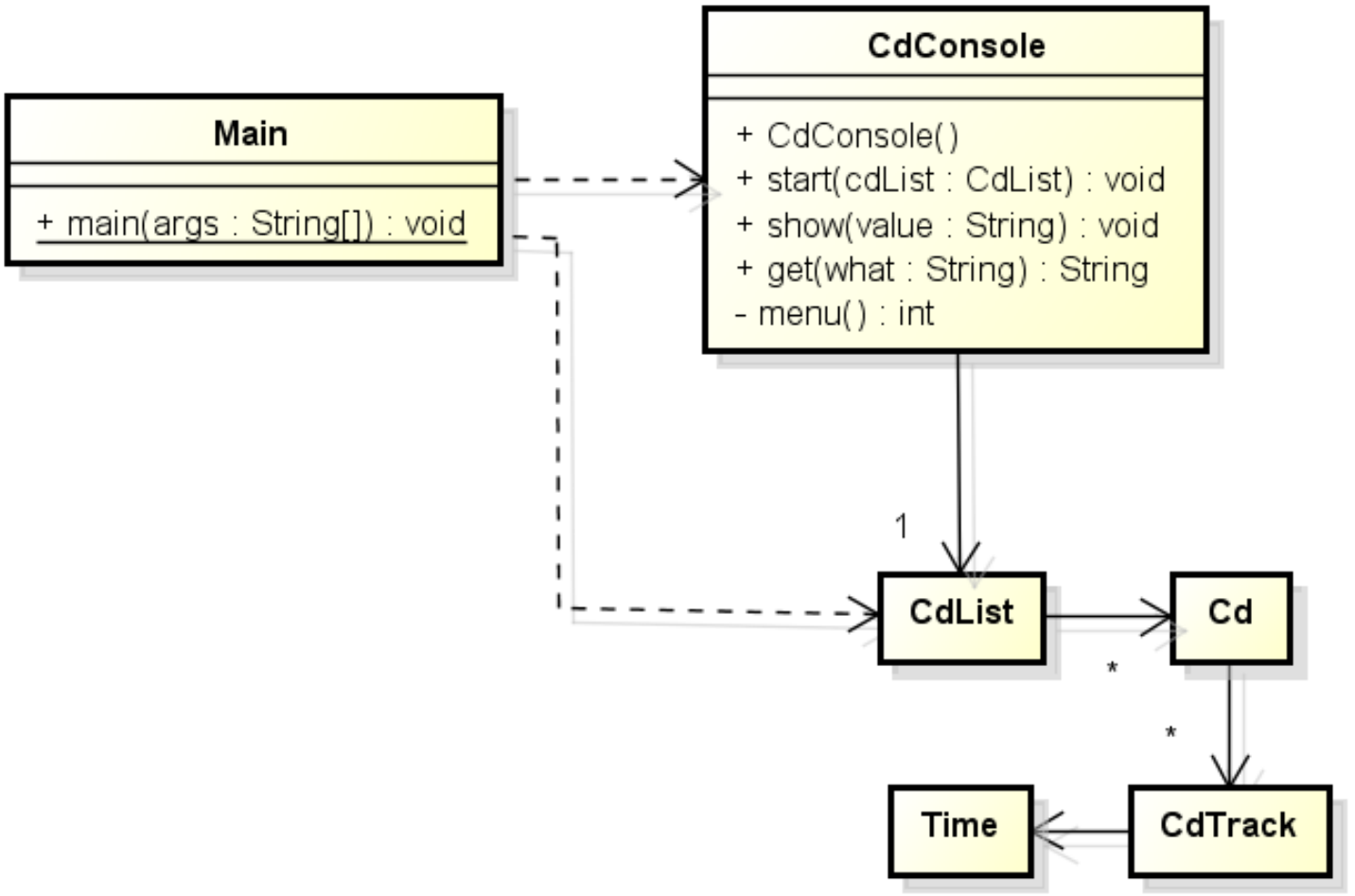
CD Application (Model)



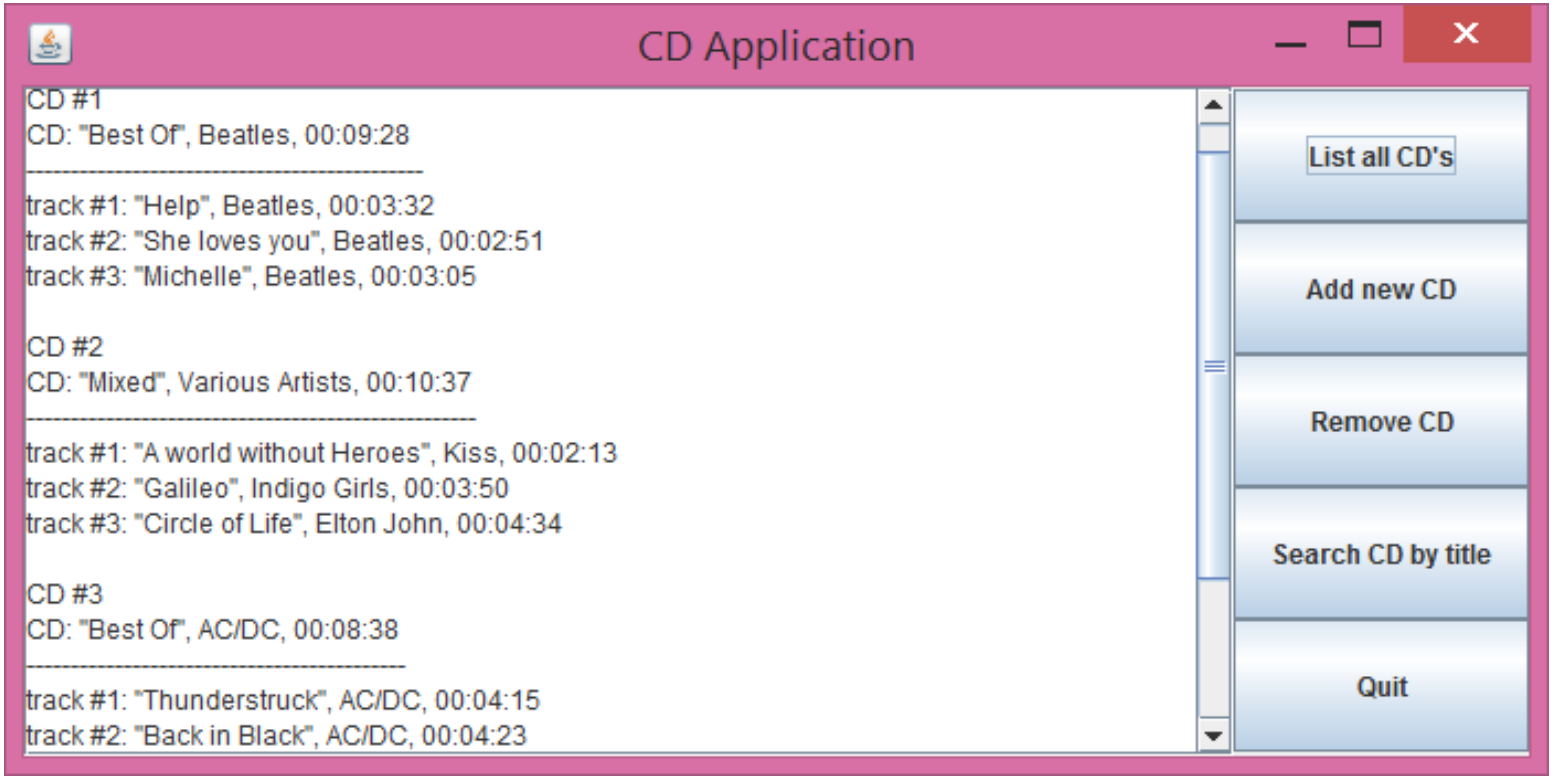
CD Application (Main)



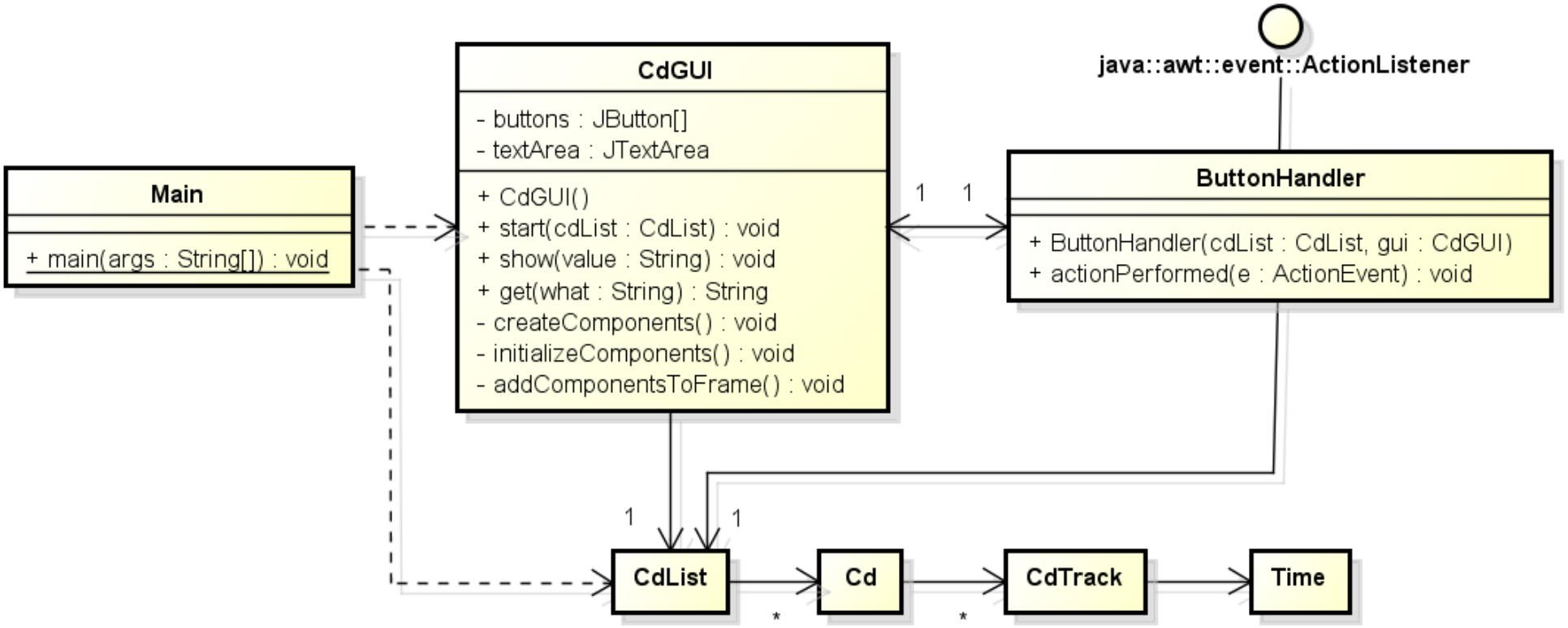
CD Application (Console)



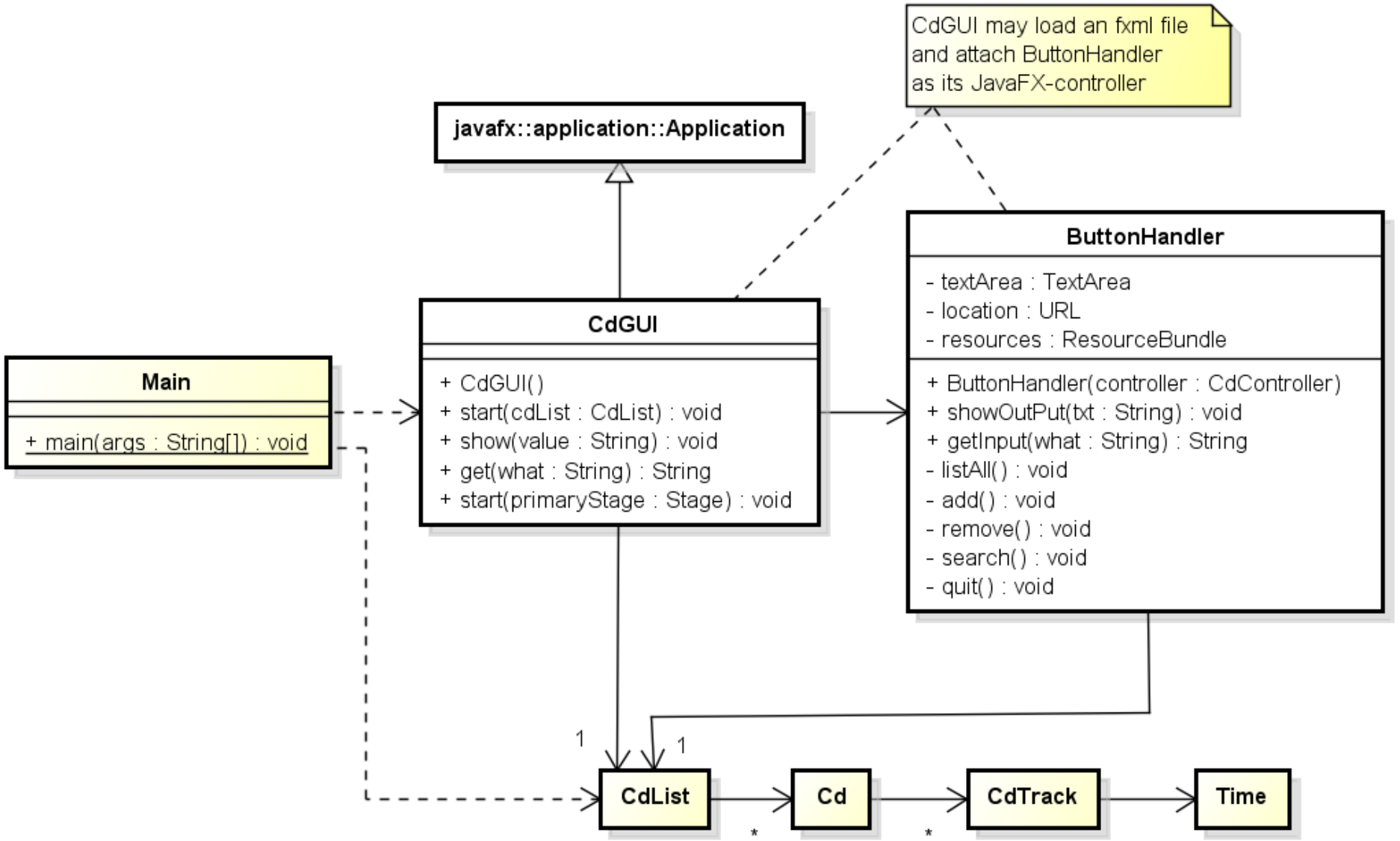
Example: CD Application (GUI)



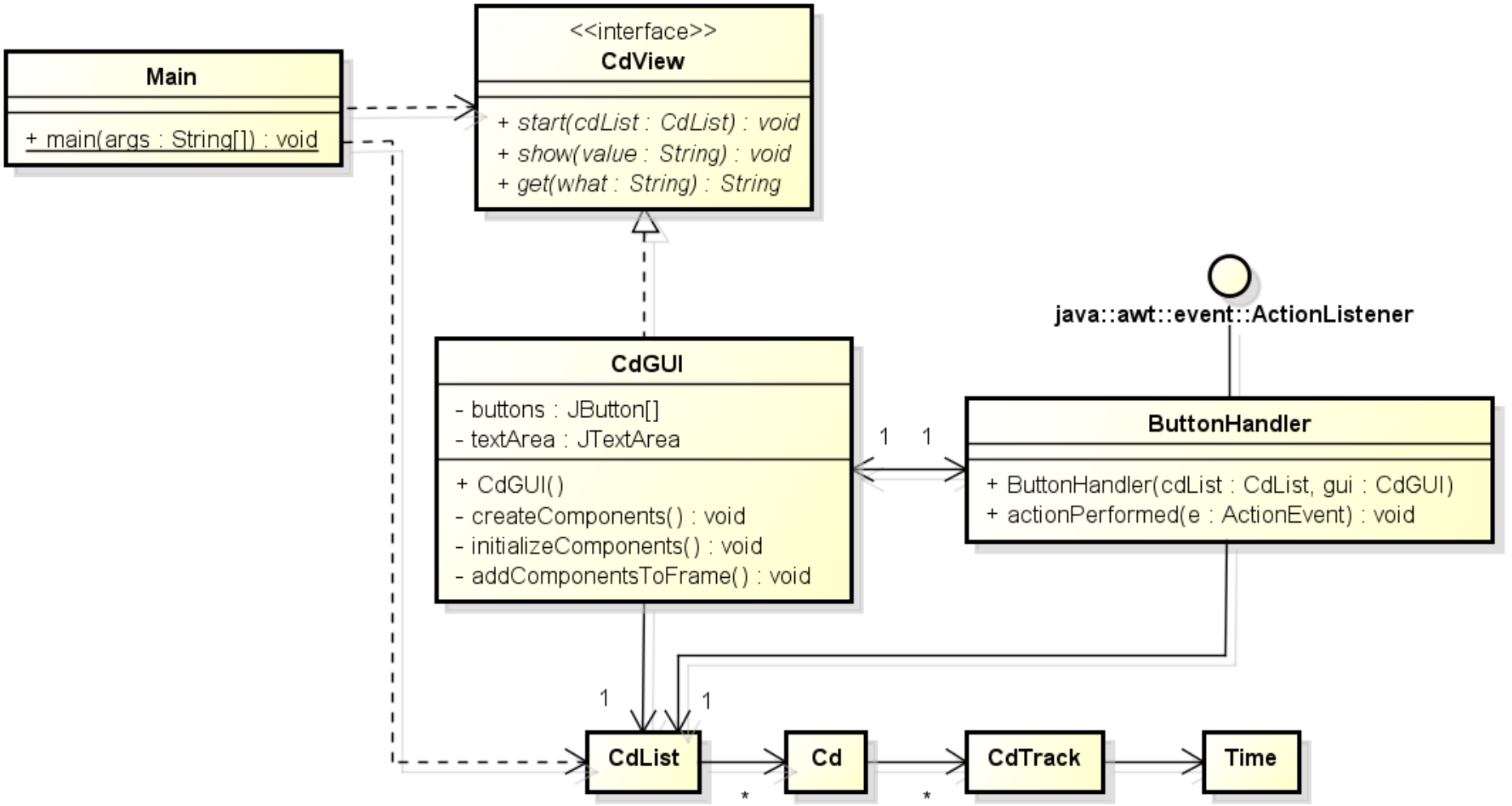
CD Application (GUI - Swing)



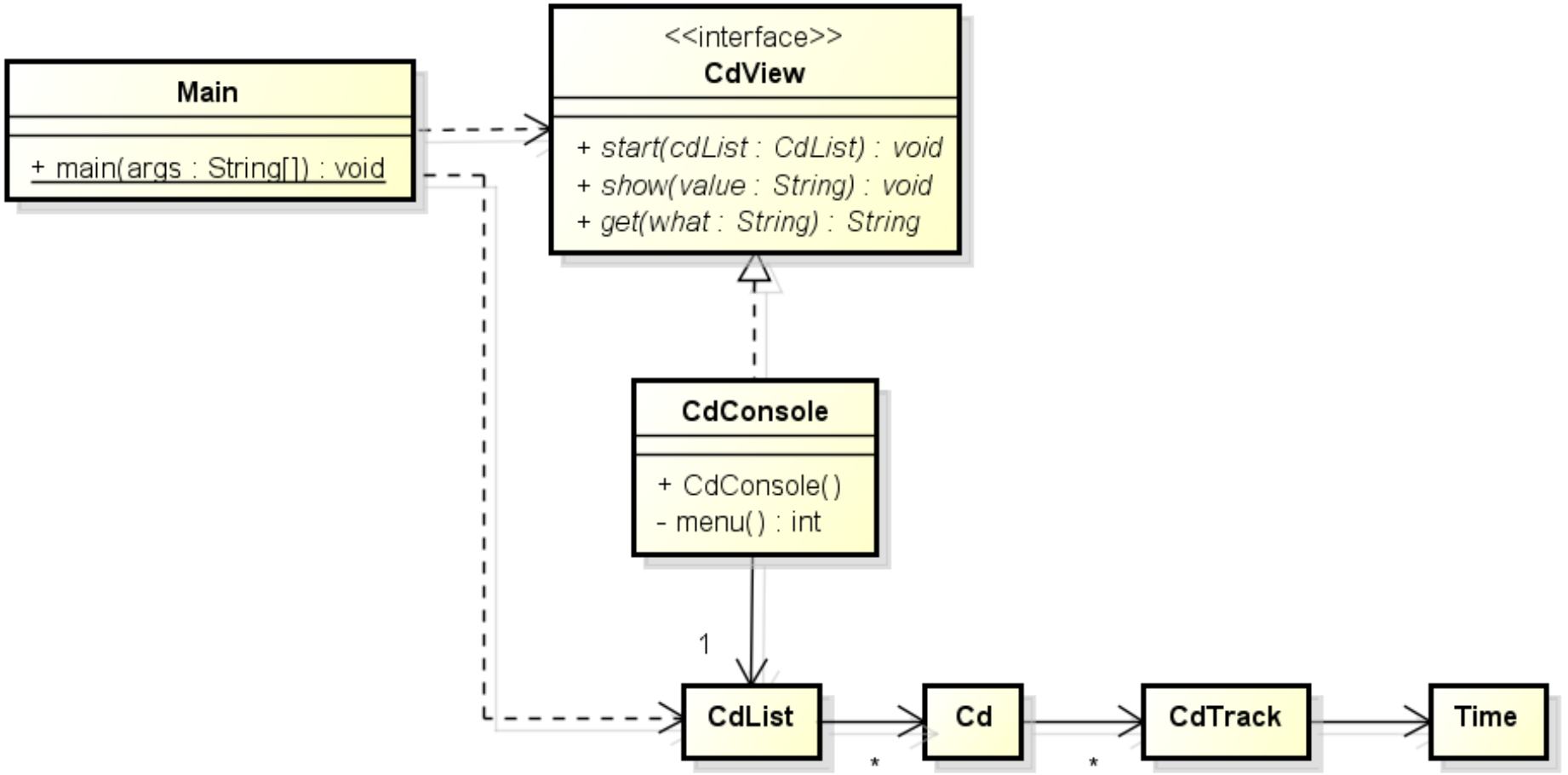
CD Application (GUI - JavaFX)



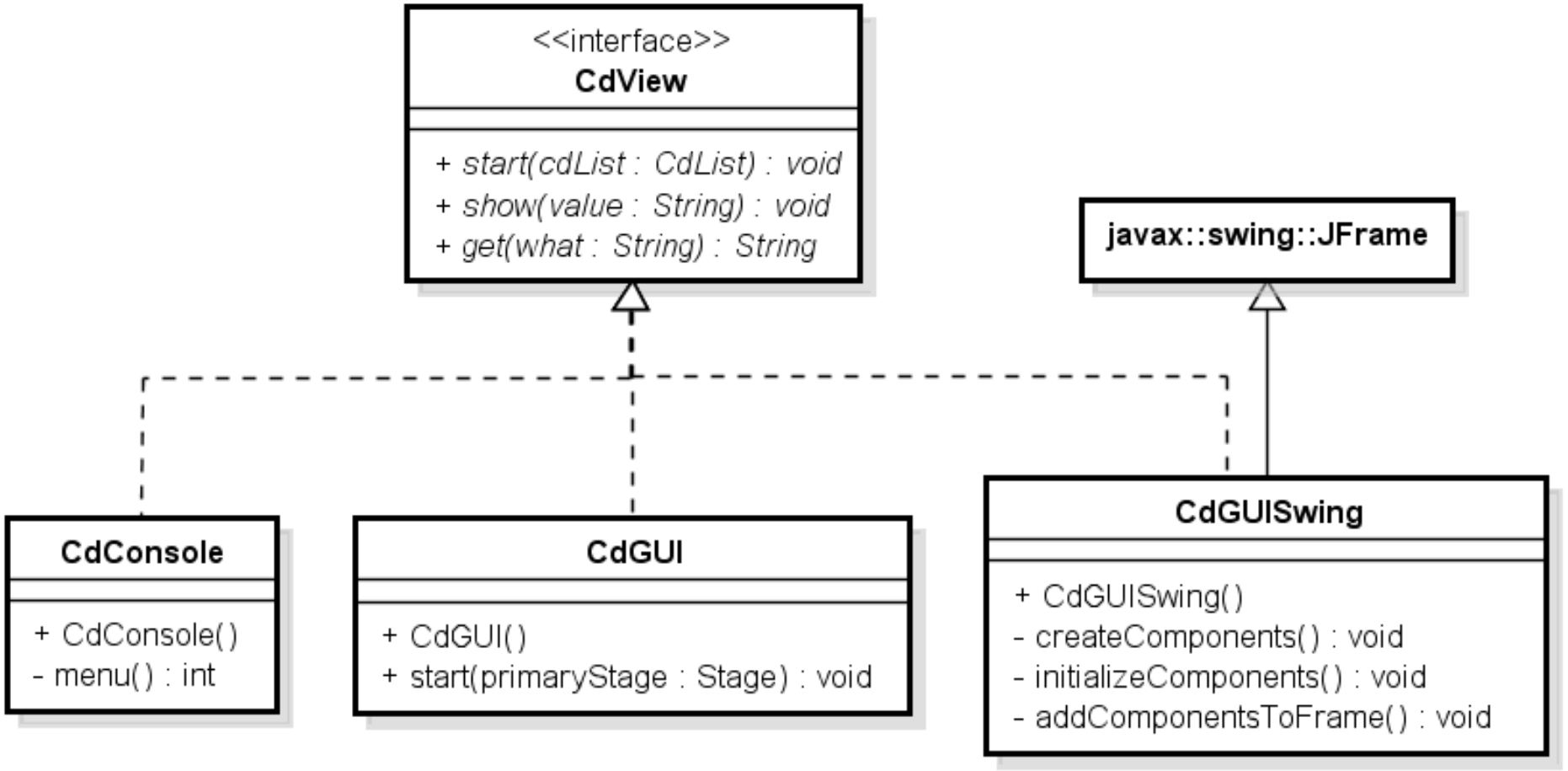
CD Application (interface)



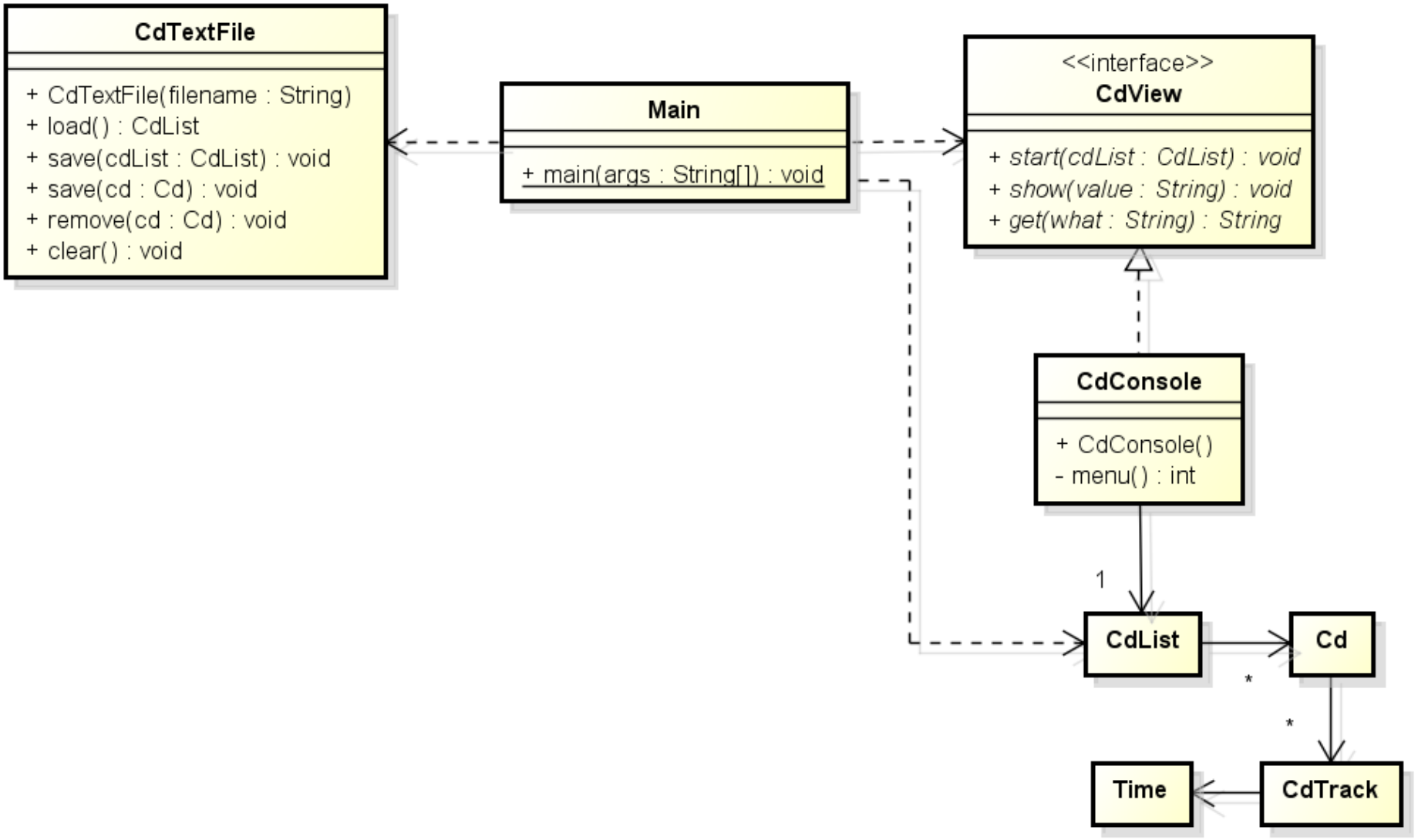
CD Application (interface)



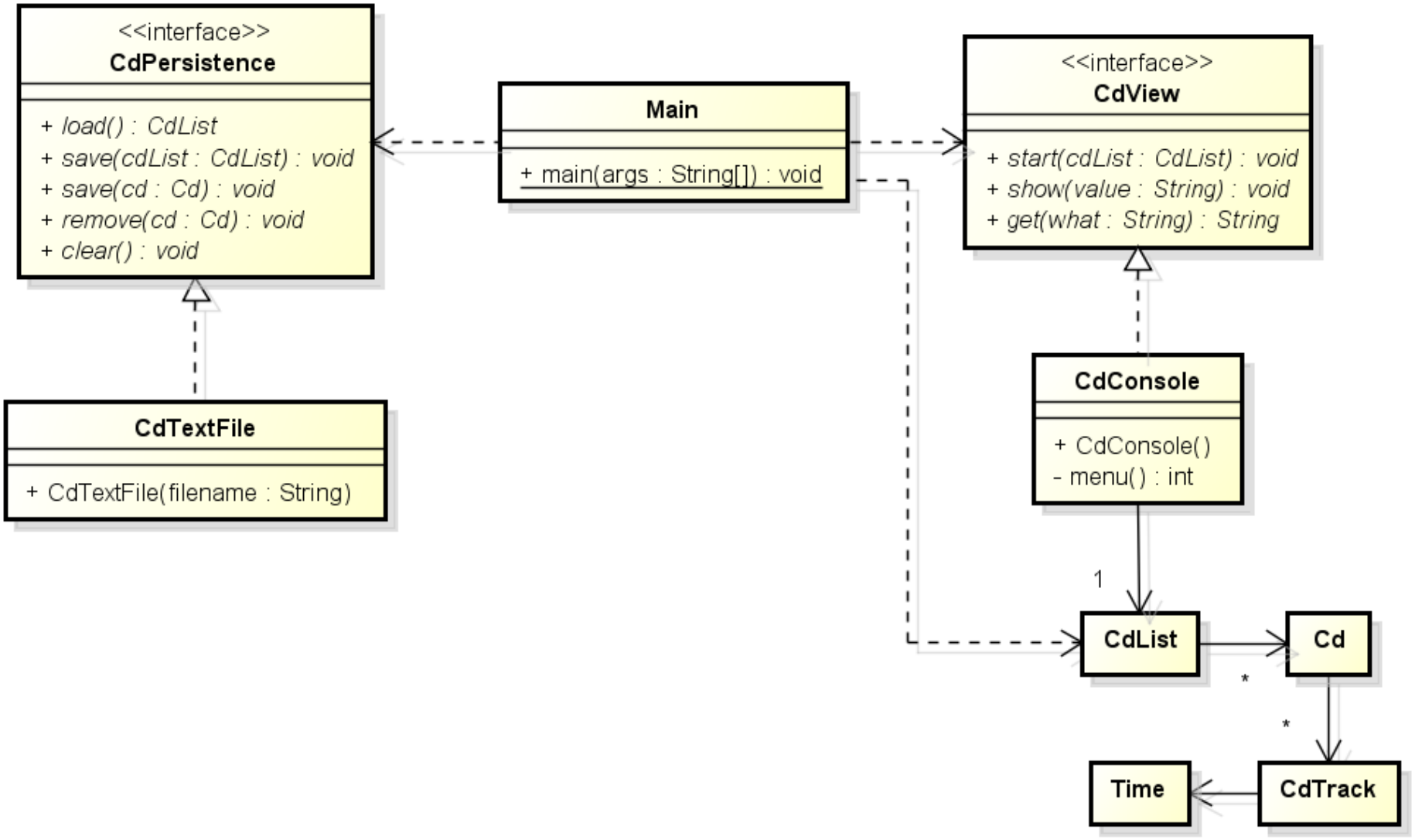
Interface CdView



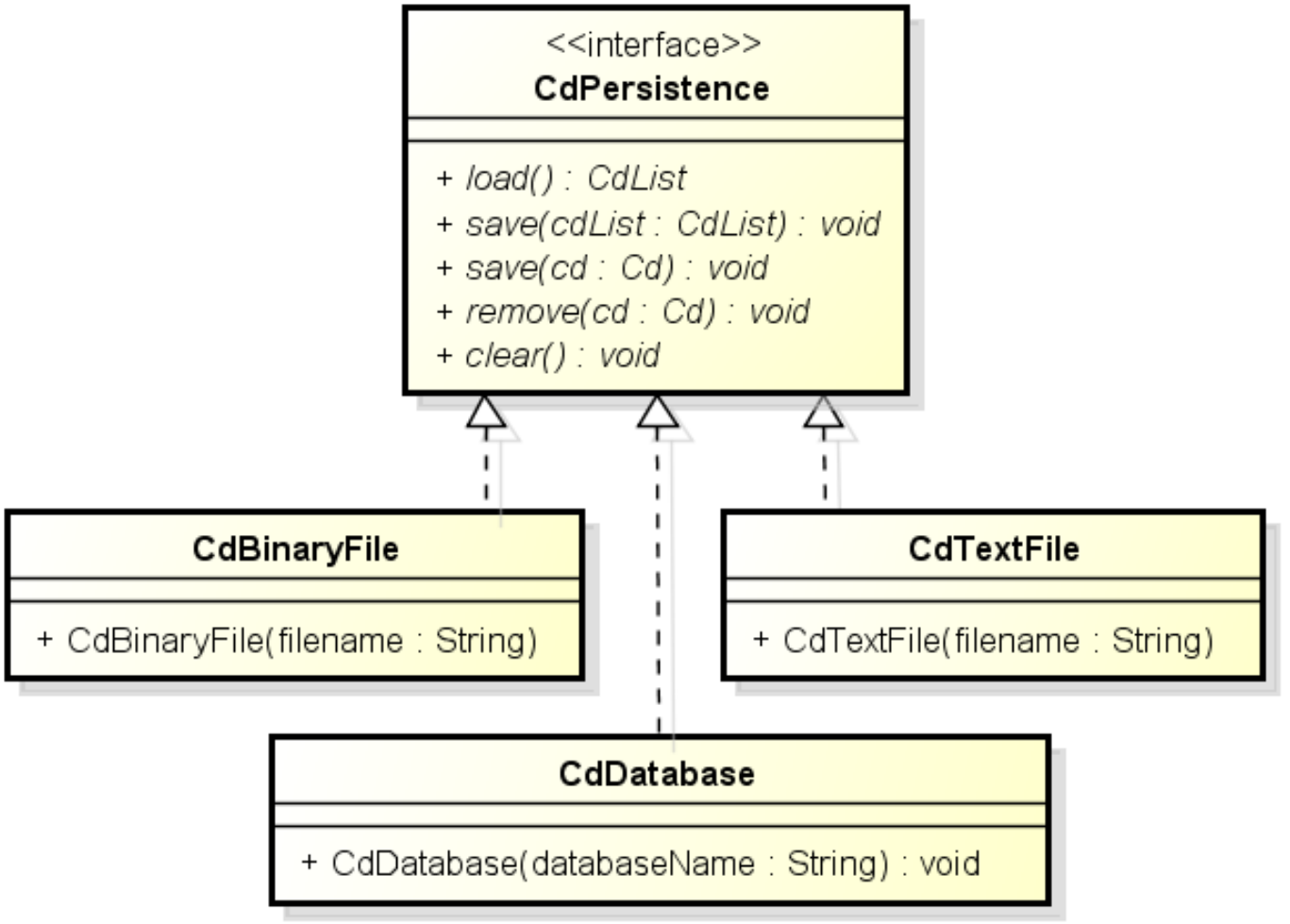
CD Application (Read from file)



CD Application (CdPersistence)



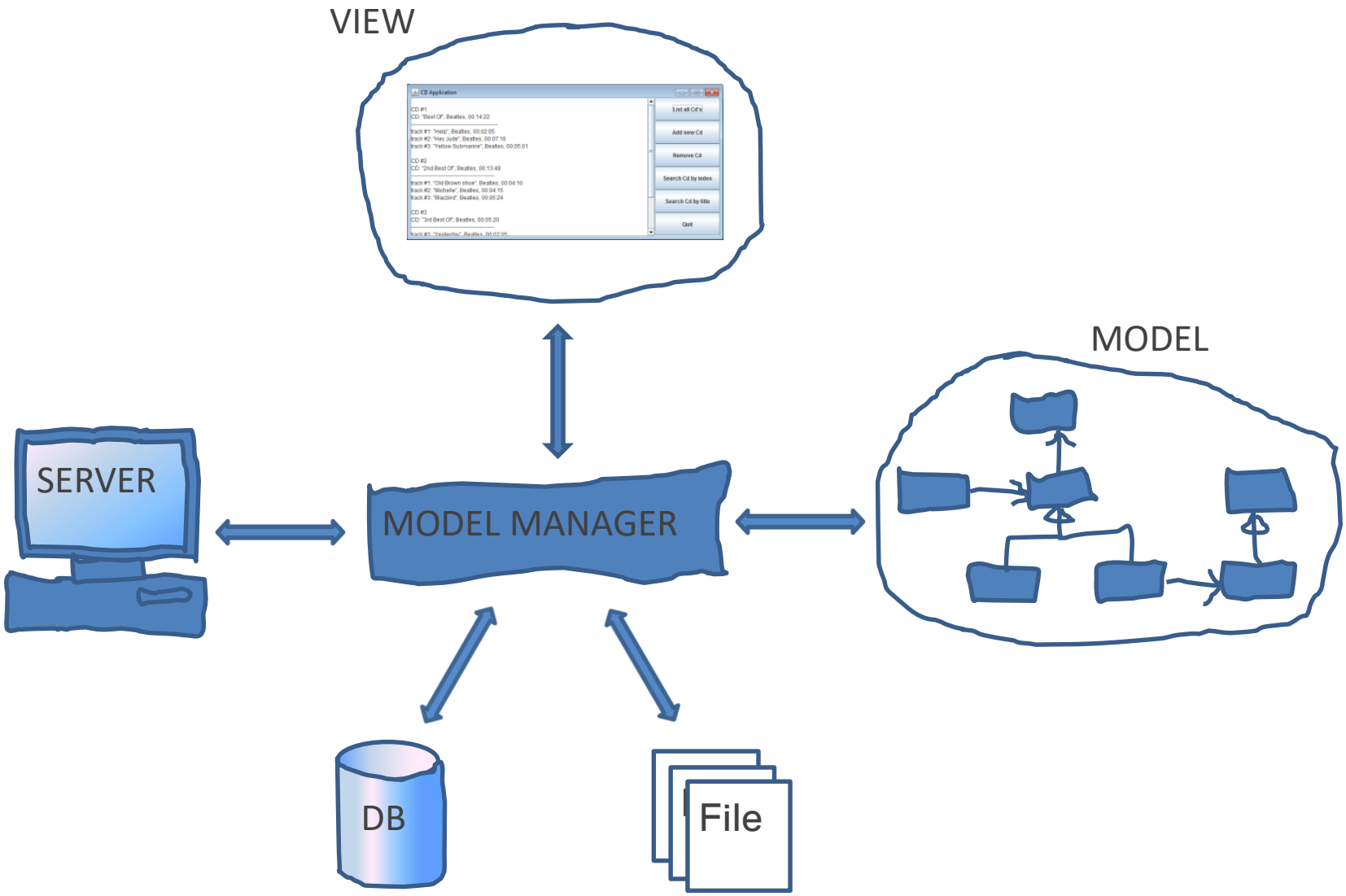
Interface CdPersistence



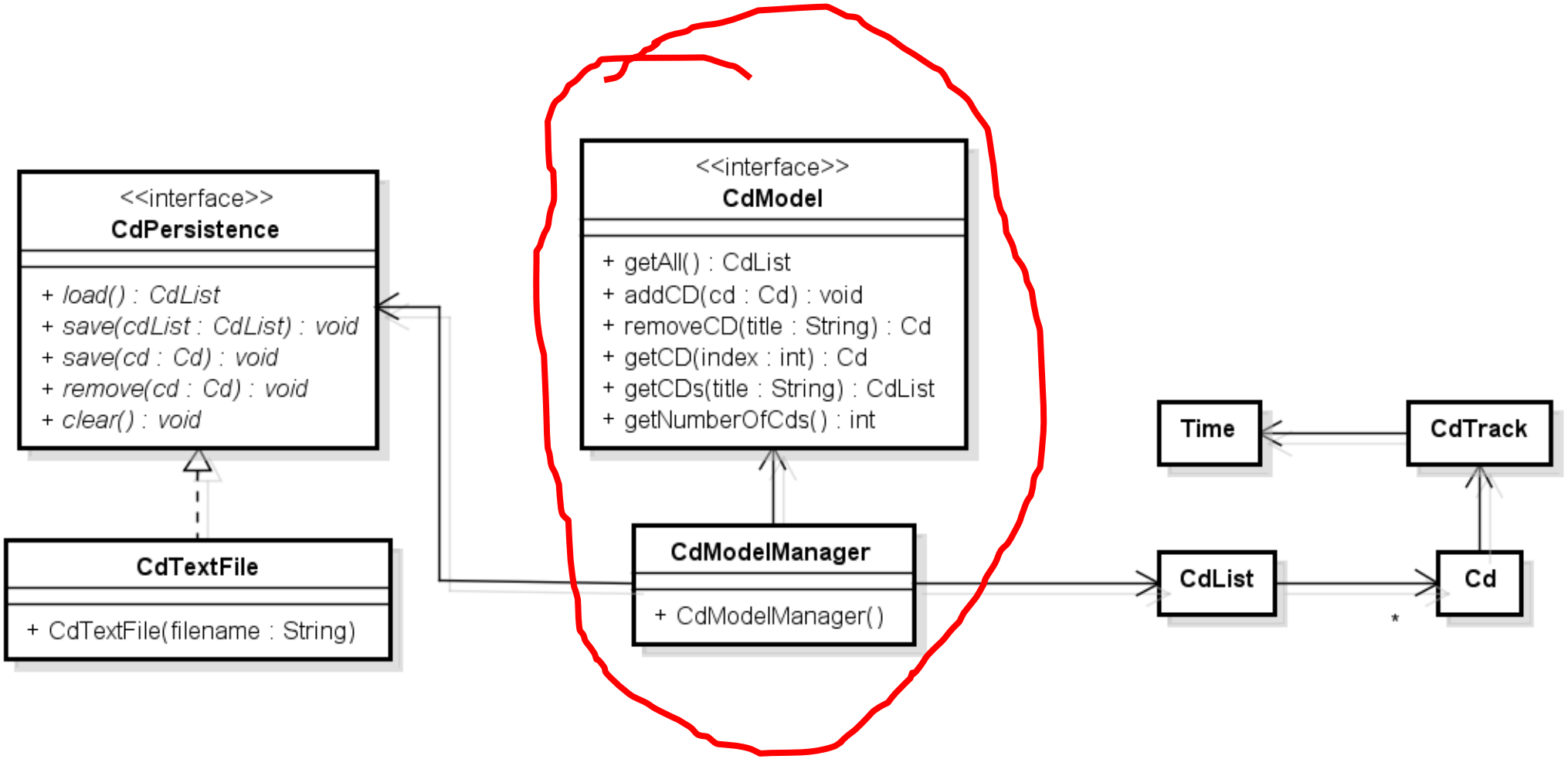
It is all about responsibility

- Model
  - Contains data and provide operations to manipulate data
- View
  - Show the model's state
  - Get input from the user
  - Take actions from the users input (a bad idea)
  - Validates data (a bad idea)
  - Modifies the model directly (a bad idea)
  - Load from and save to file or database (a bad idea)

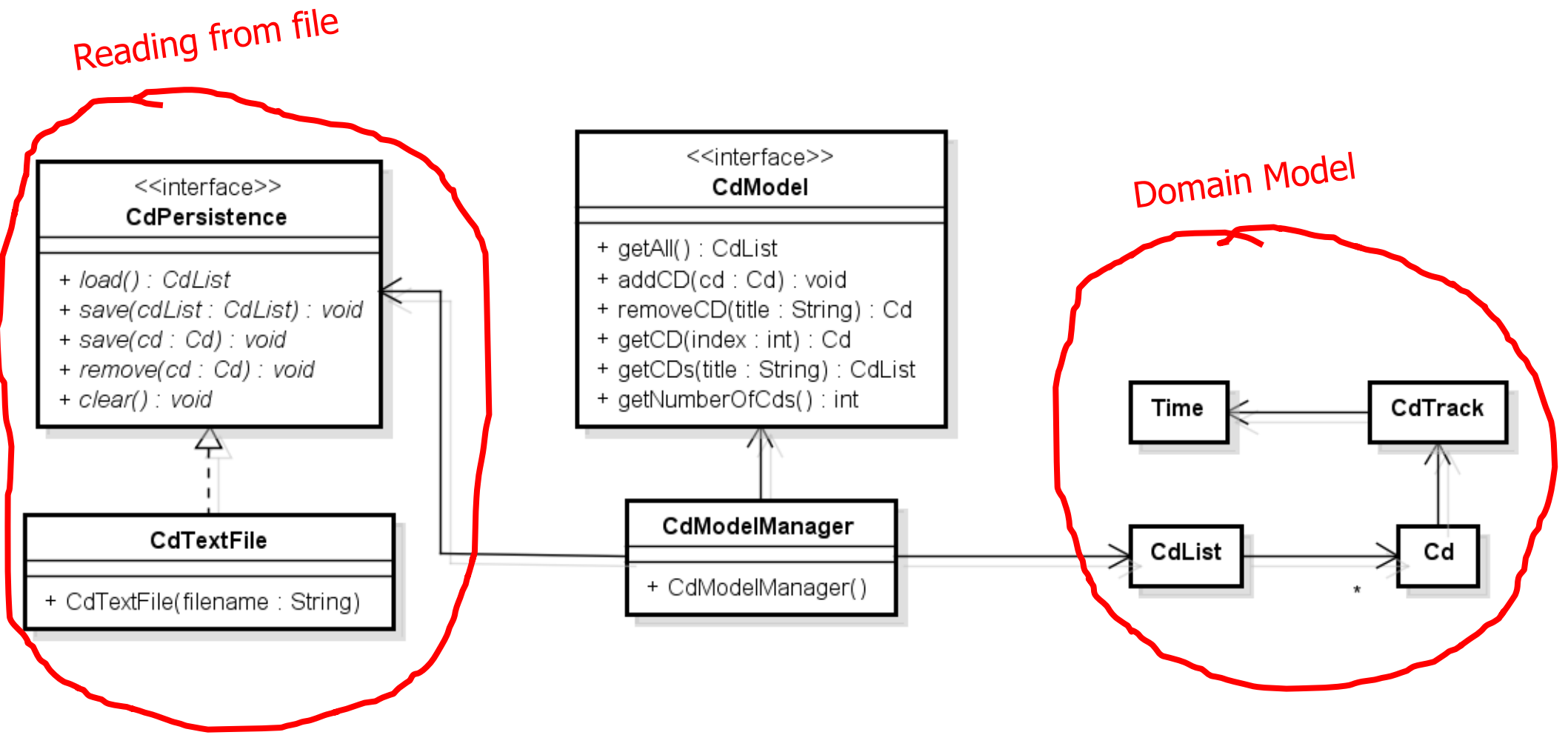
General System Overview



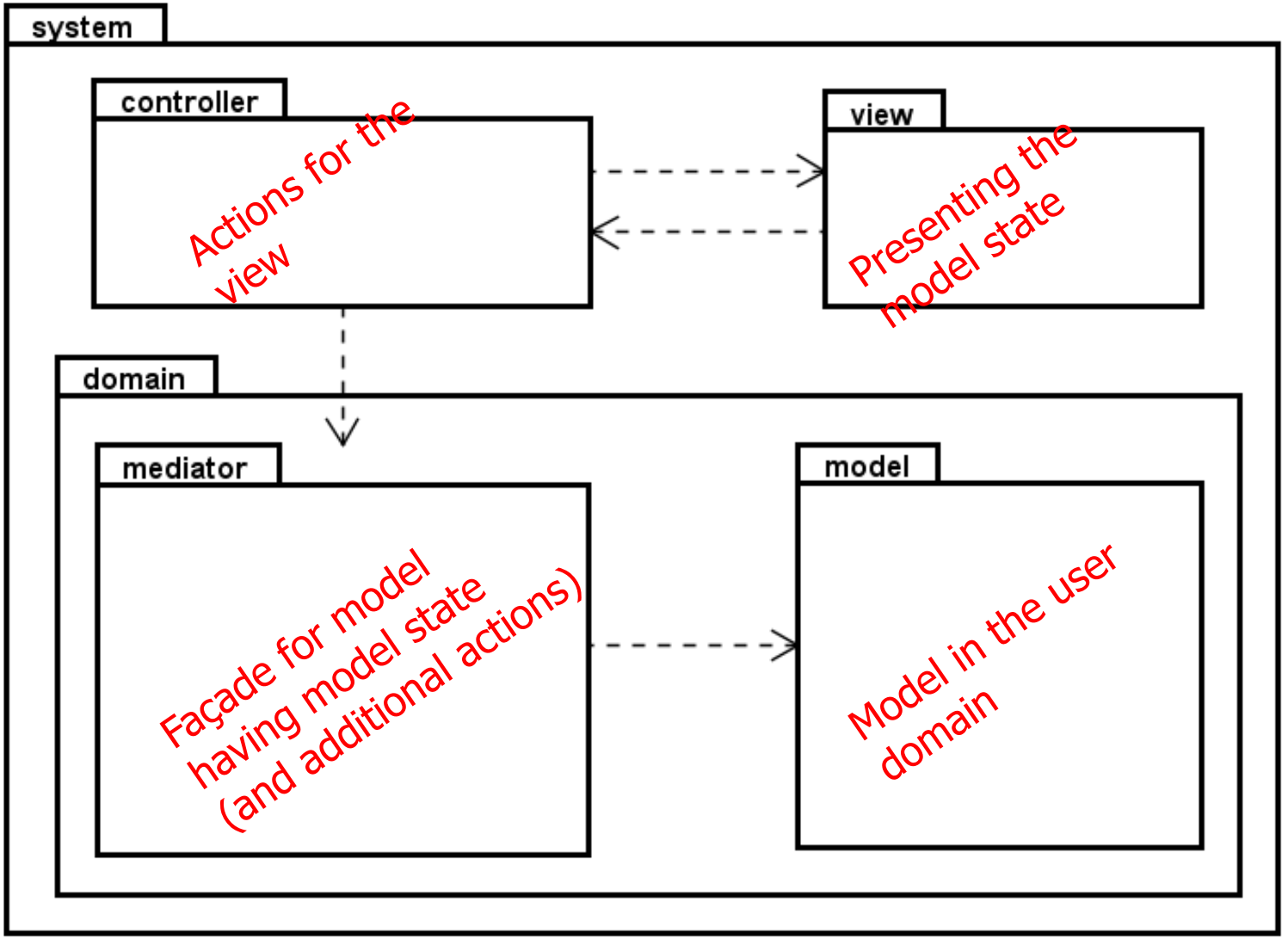
CdModel (Model Manager)



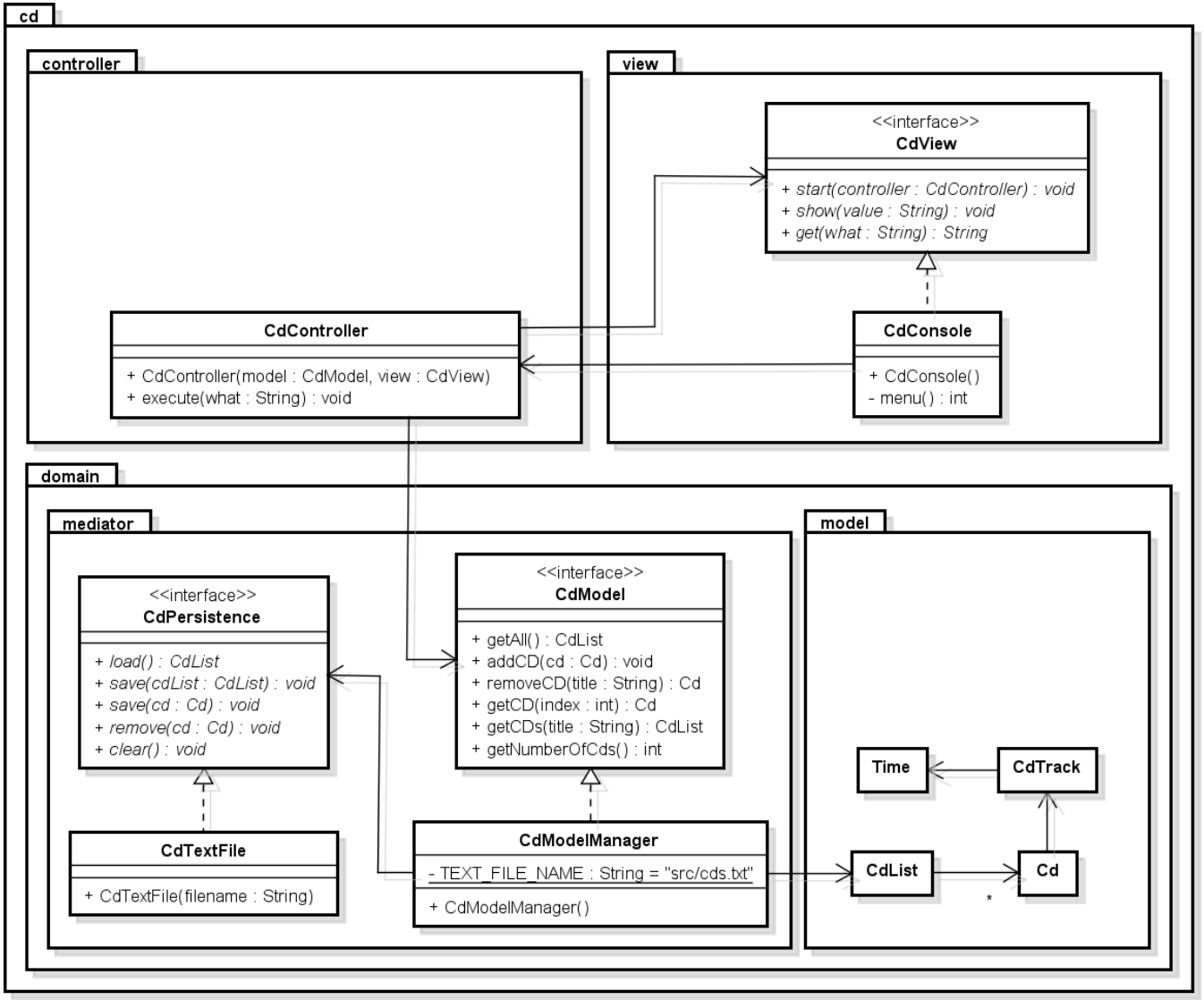
Model Manager responsibility



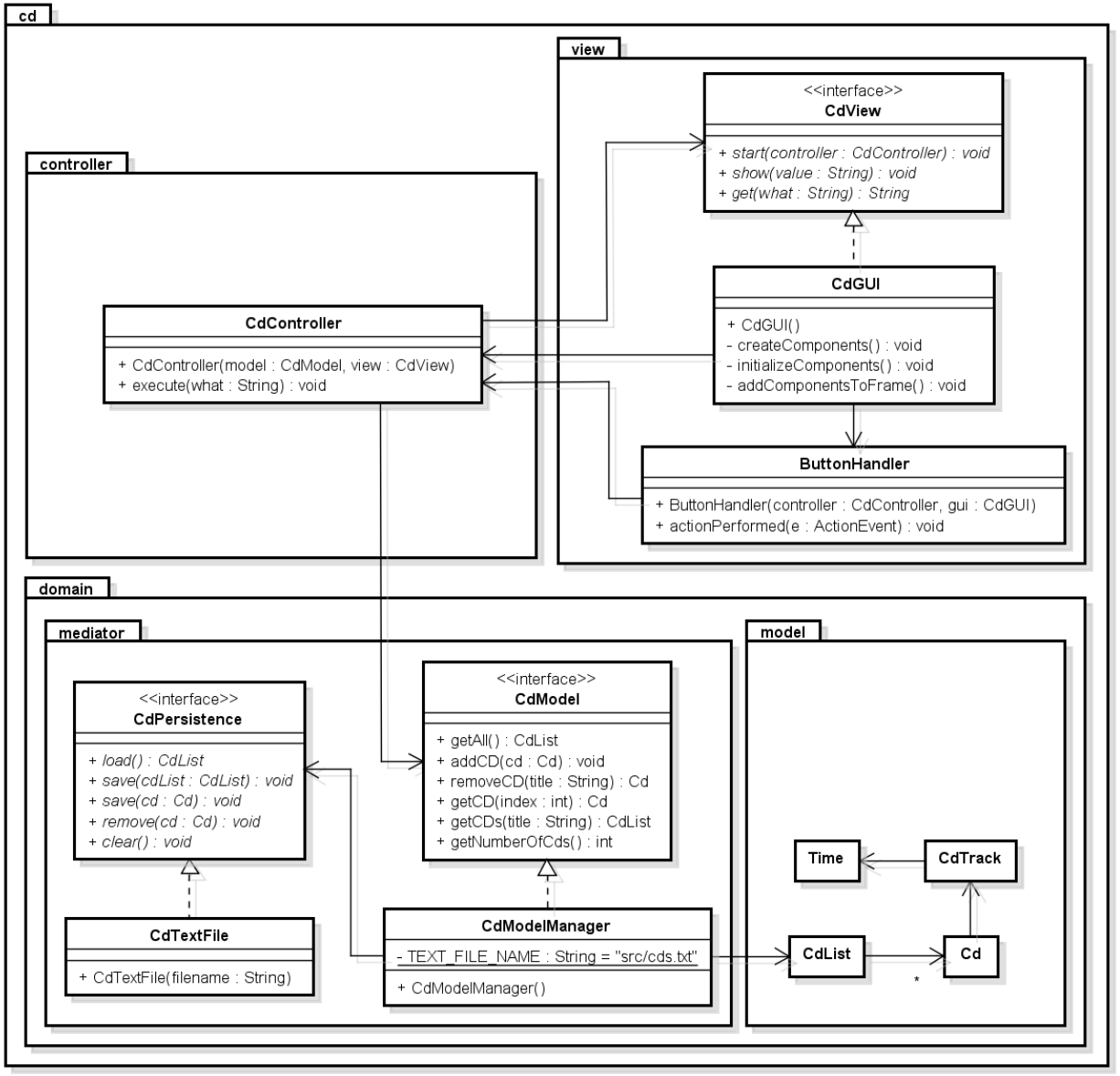
Model-View-Controller (MVC)



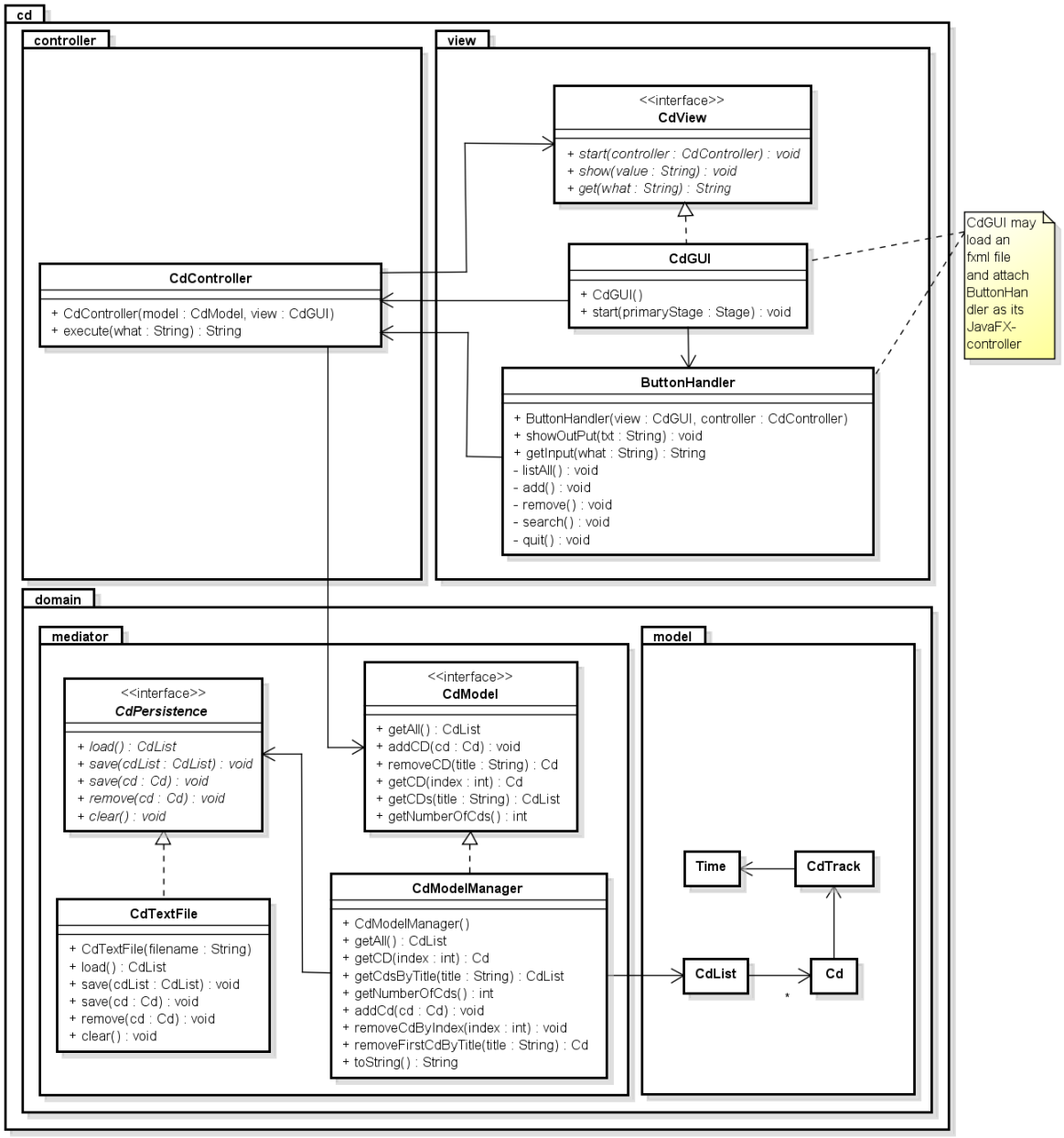
MVC (Console application)



MVC (GUI Application - Swing)

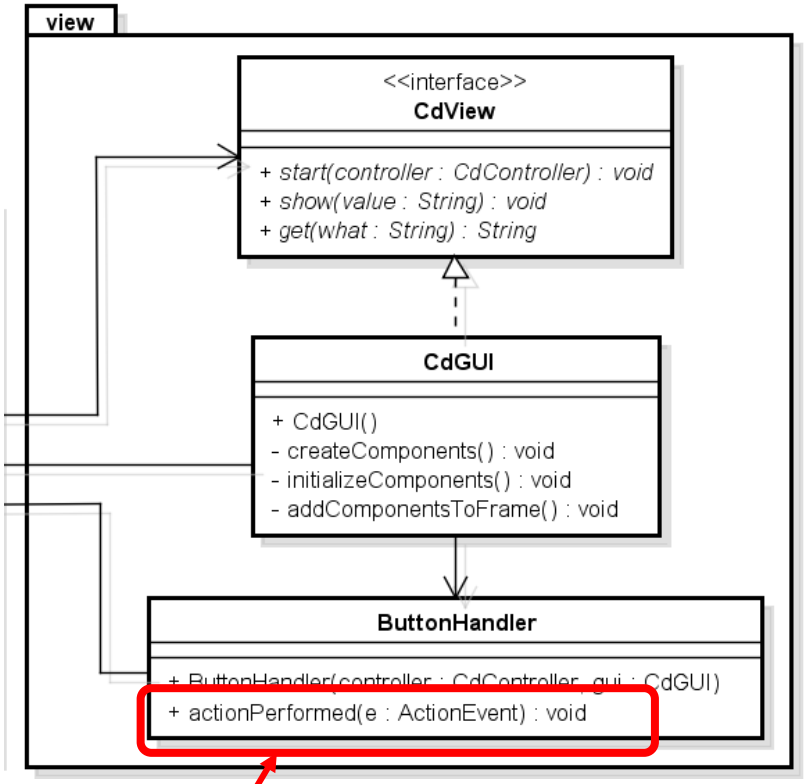


MVC (GUI Application - JavaFX)



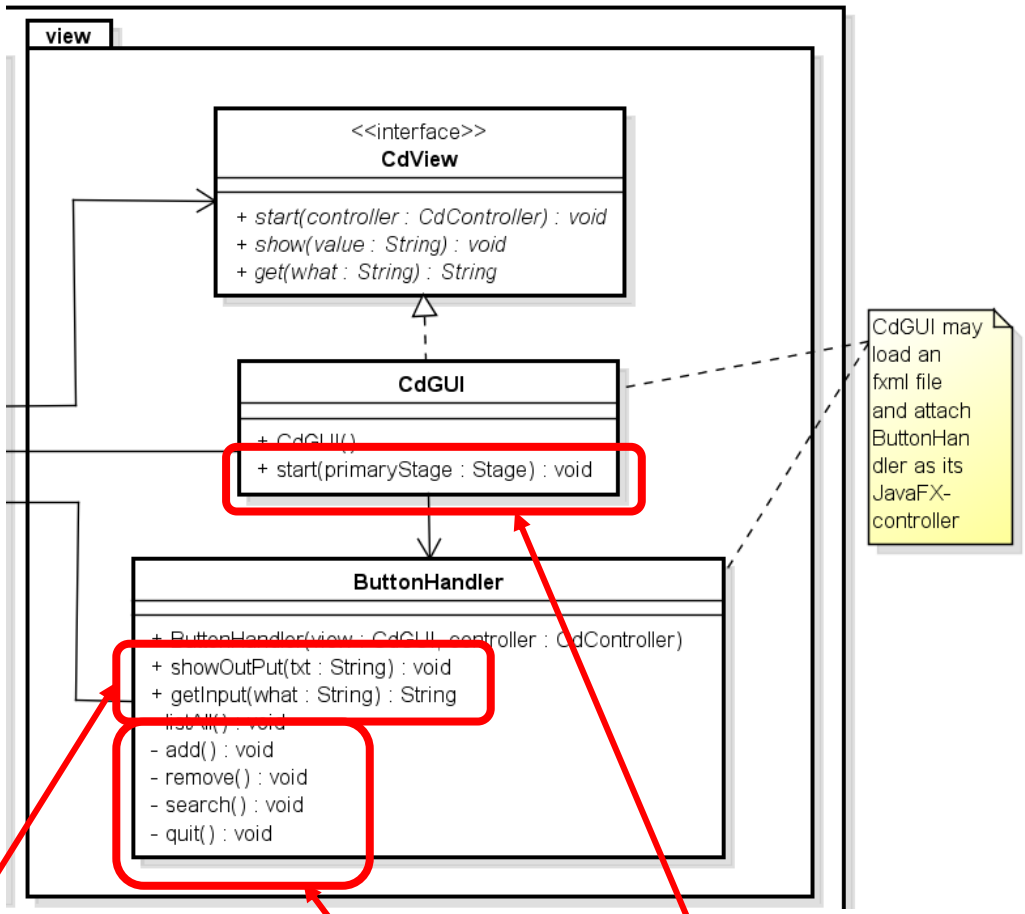
MVC (GUI Application – Swing vs JavaFX)

Swing



Implementing ActionListener

JavaFX

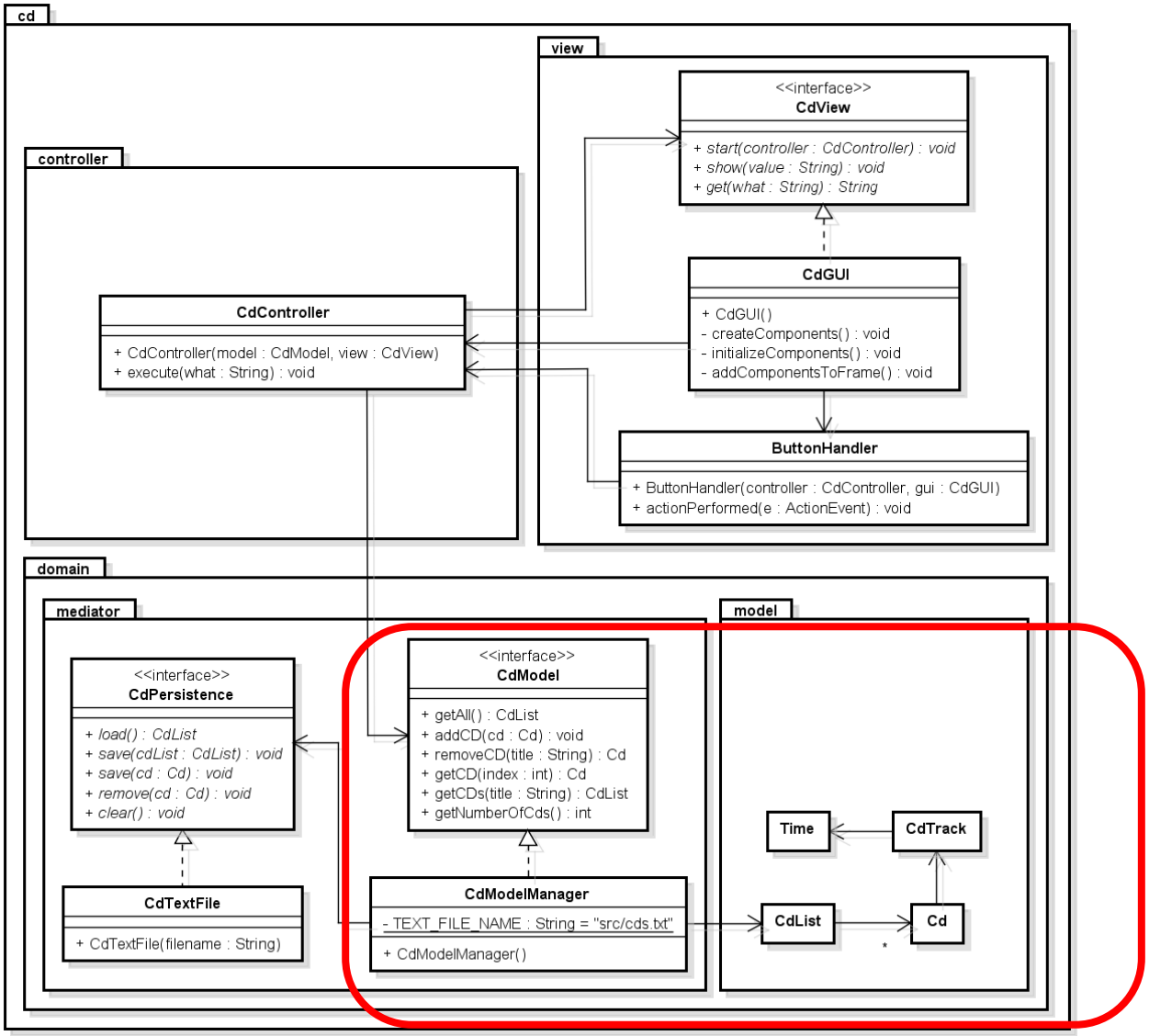


Extending Application

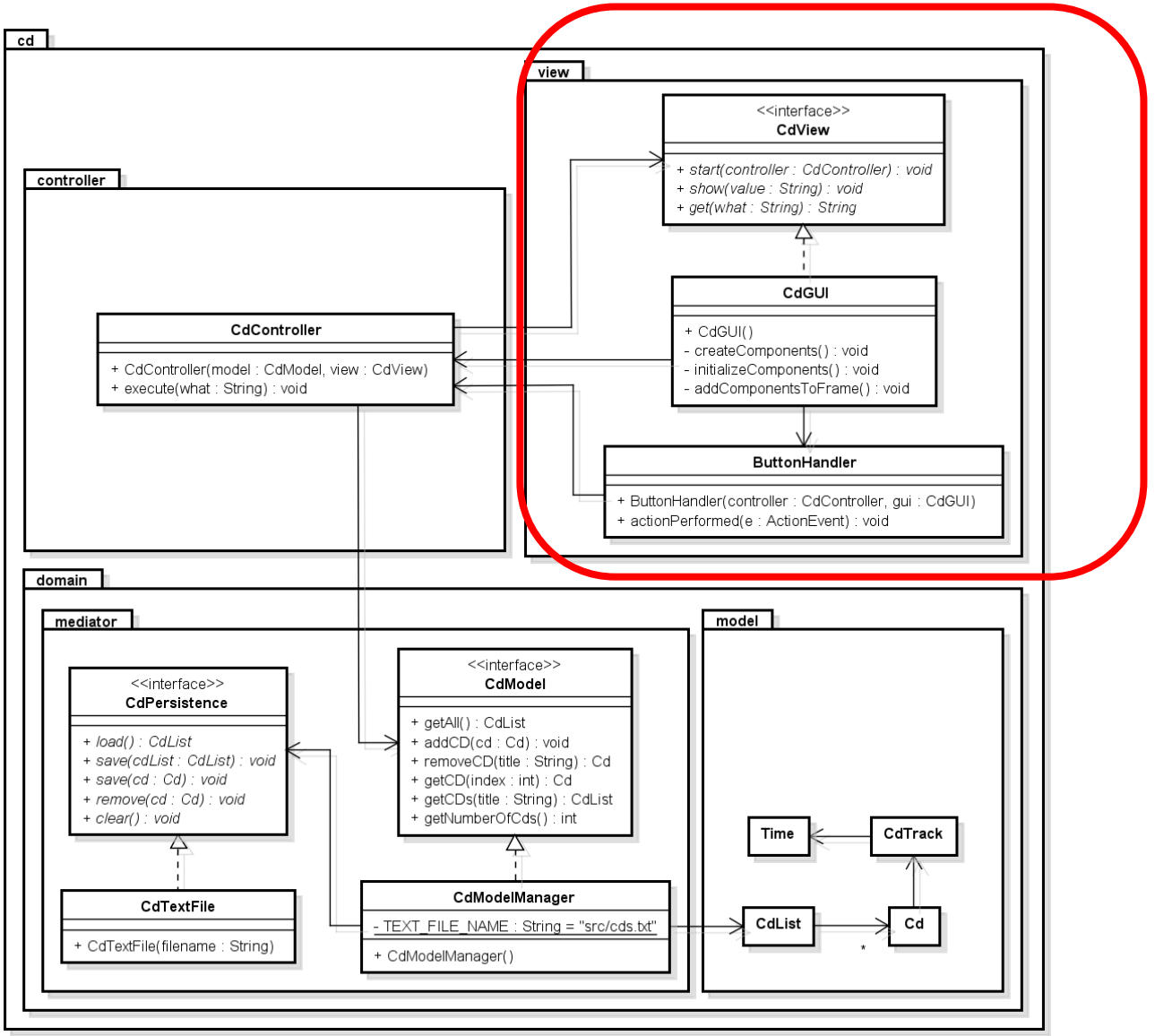
Methods to get and show

Methods called from buttons (defined in the FXML file)

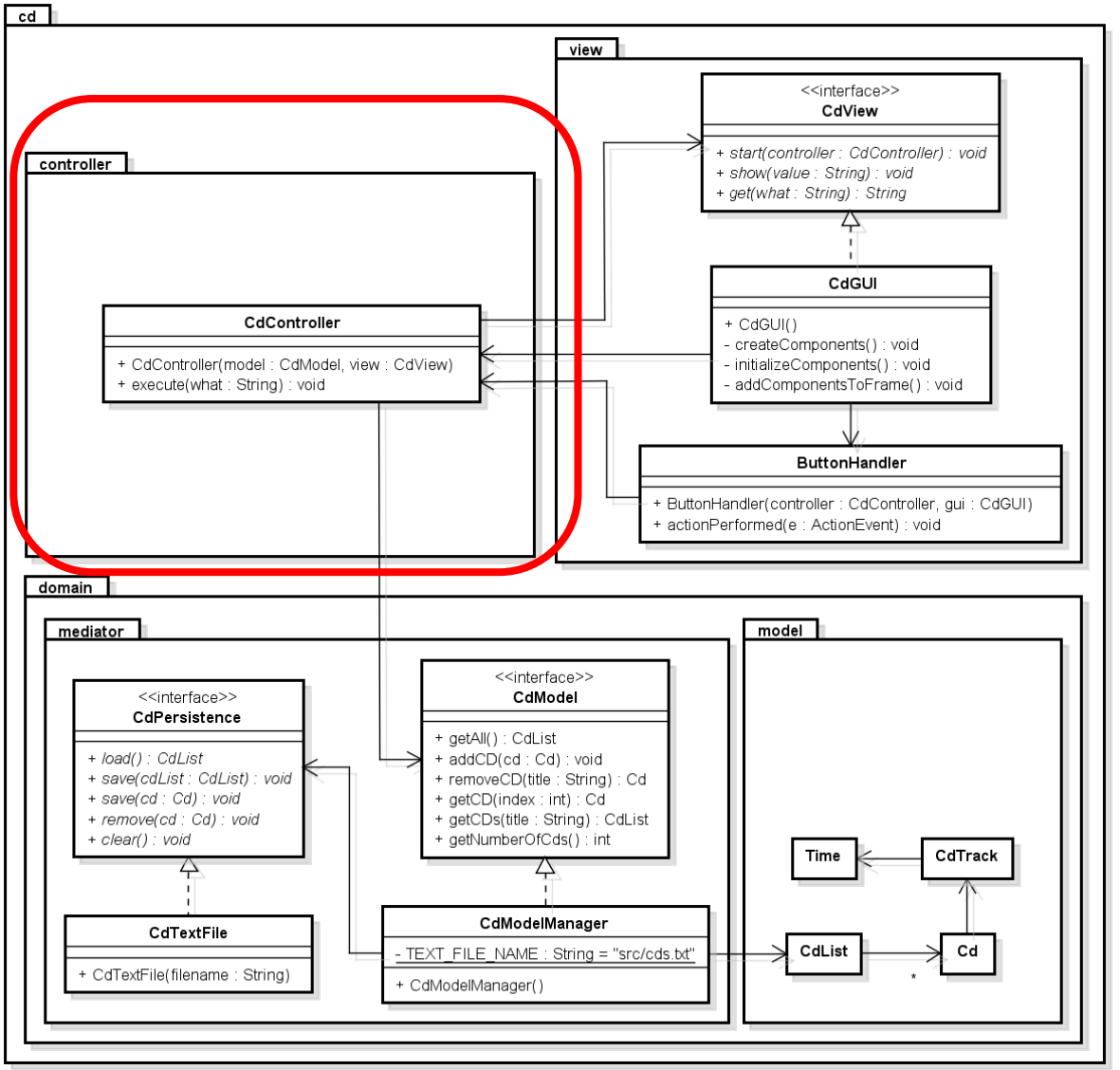
Model-View-Controller



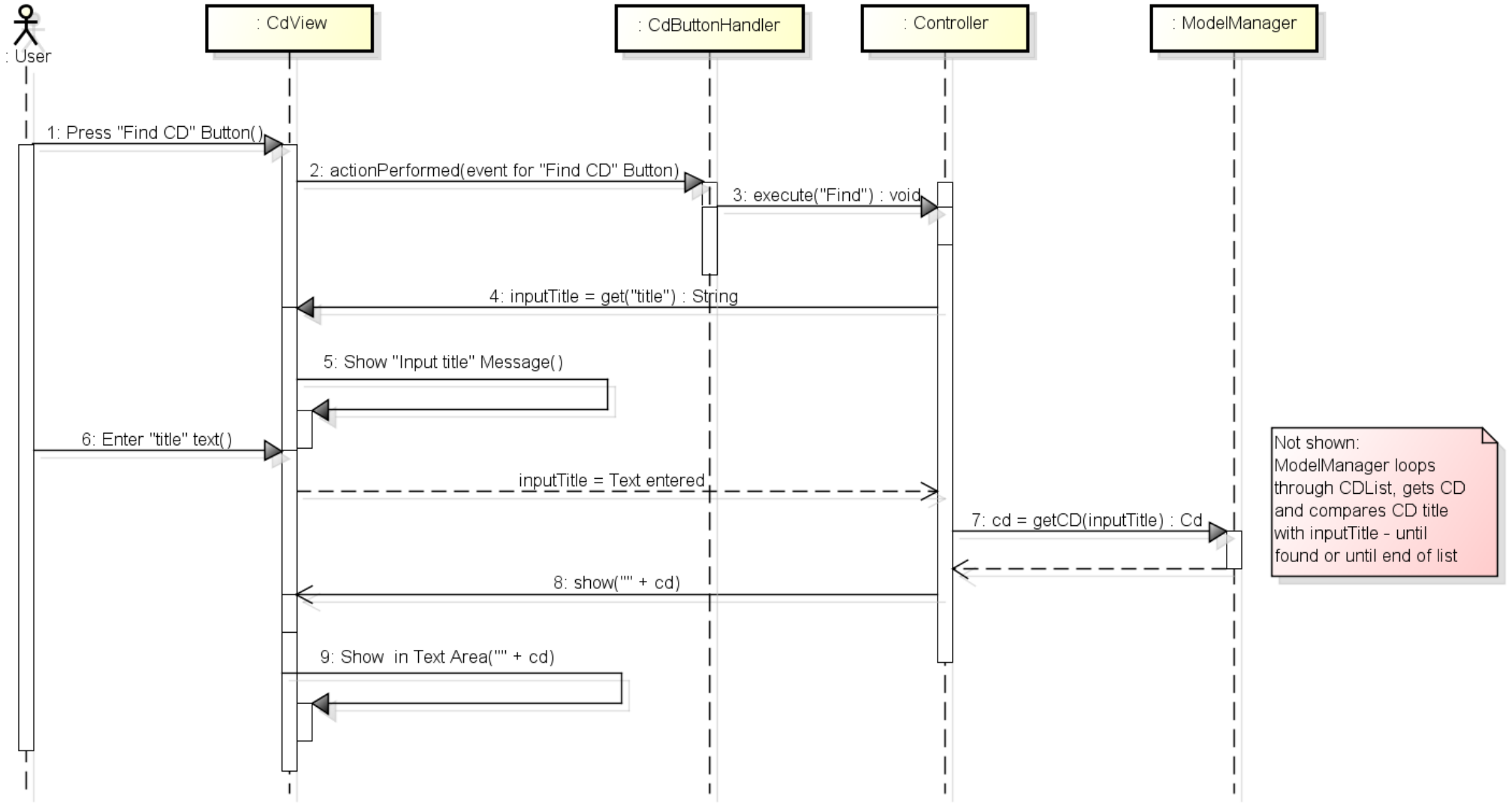
Model-View-Controller



Model-View-Controller

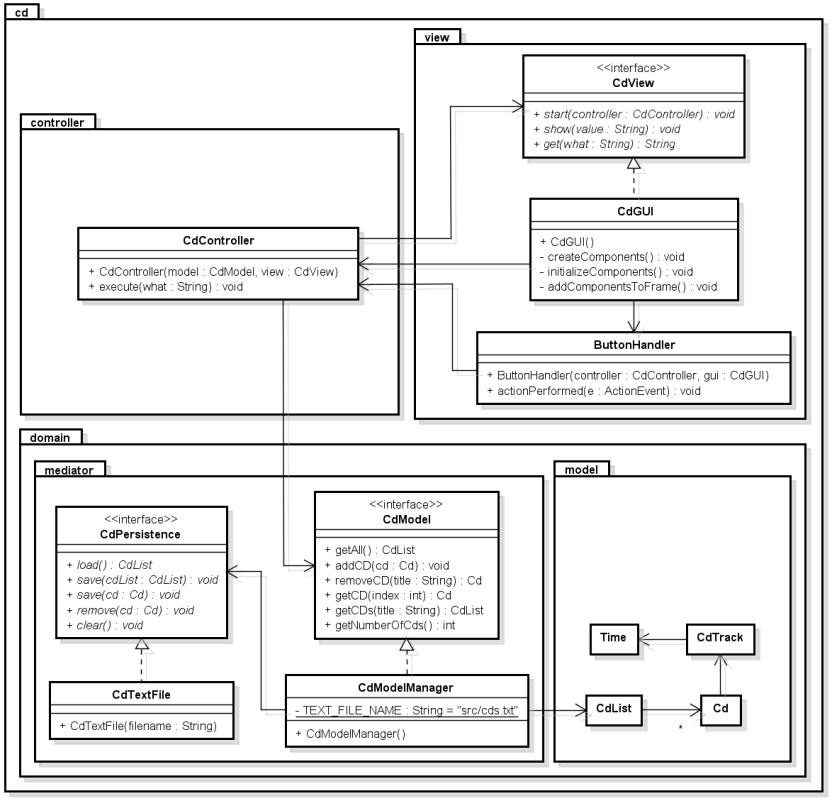


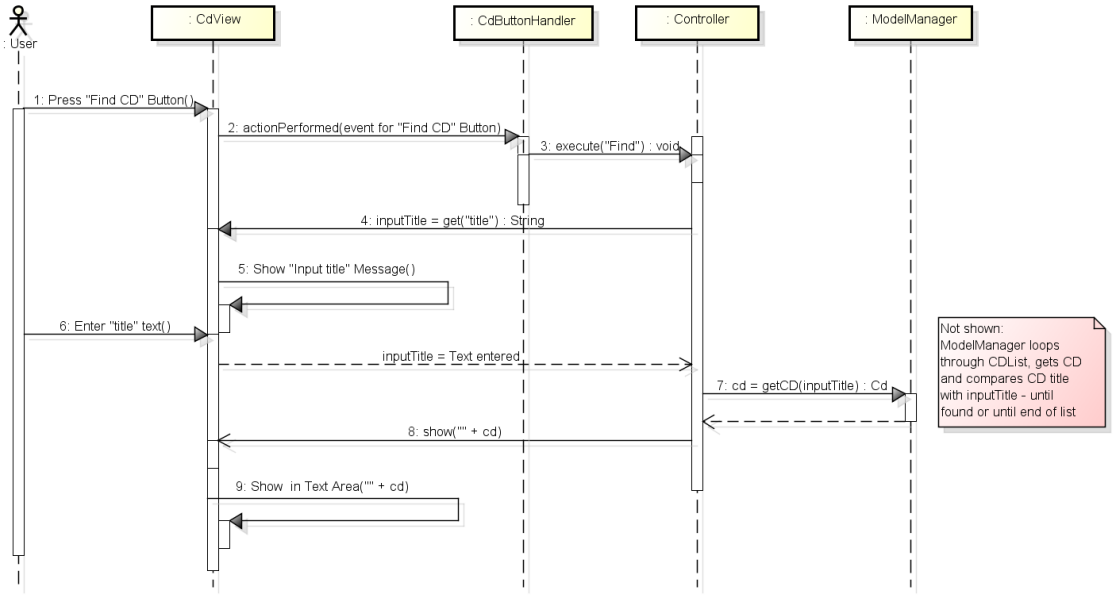
Example: Find CD by title



Model-View-Controller (documentation)

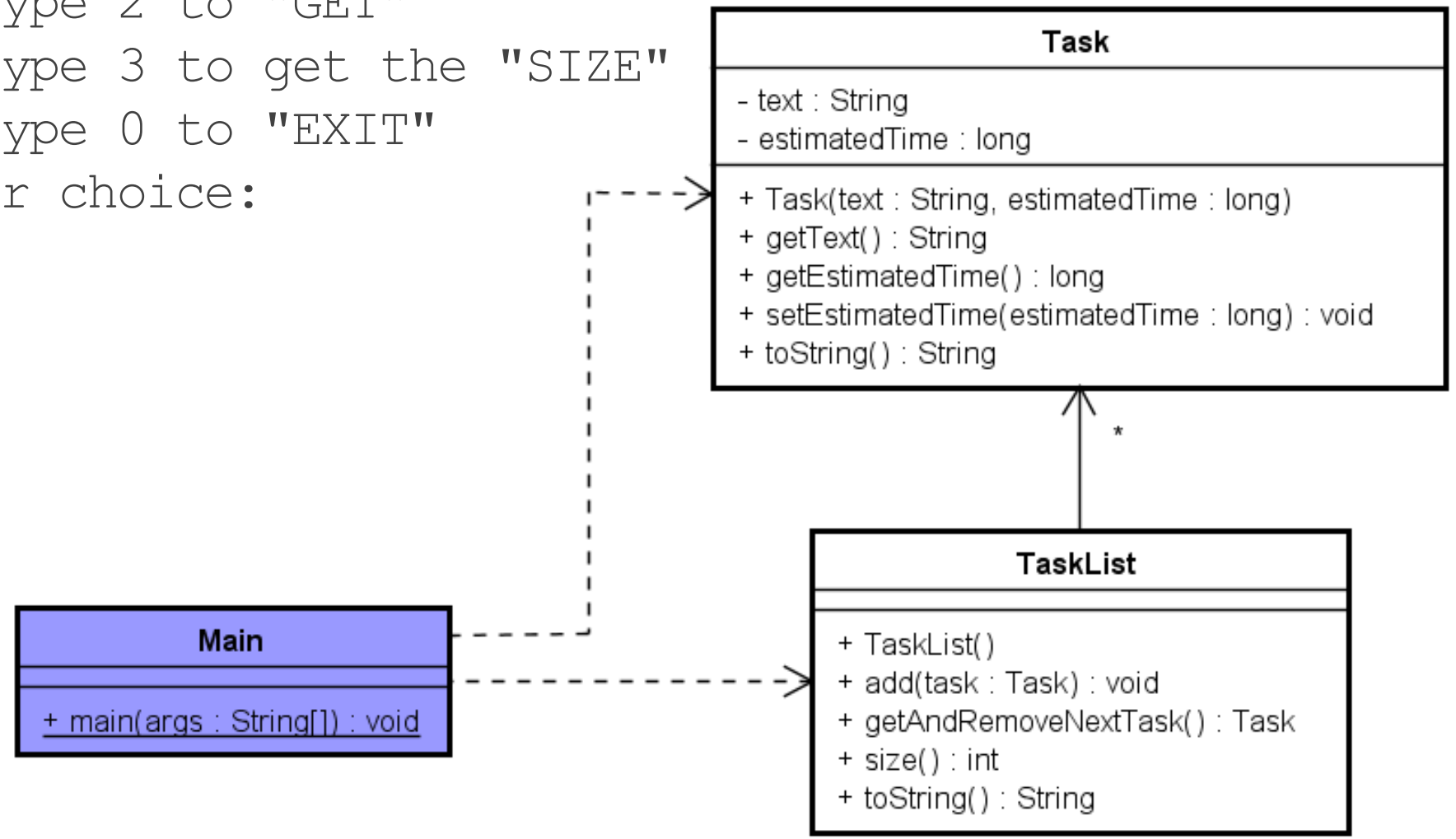
- Class diagram
- Sequence diagram to prove that you follow the MVC pattern
- Description





Another example – a task list

```
1) Type 1 for "ADD"
2) Type 2 to "GET"
3) Type 3 to get the "SIZE"
0) Type 0 to "EXIT"
Enter choice:
```



Task List – Main (1/3)

```
public static void main(String args[])
{
   TaskList taskList = new TaskList();
   Scanner input = new Scanner(System.in);

   boolean continueWorking = true;
   while (continueWorking)
   {
      System.out.print("1) Type 1 for \"ADD\"\n"
            + "2) Type 2 to \"GET\""
            + "\n3) Type 3 to get the \"SIZE\"\n"
            + "0) Type 0 to \"EXIT\"\nEnter choice: ");
      int choice = input.nextInt();
      input.nextLine();
```

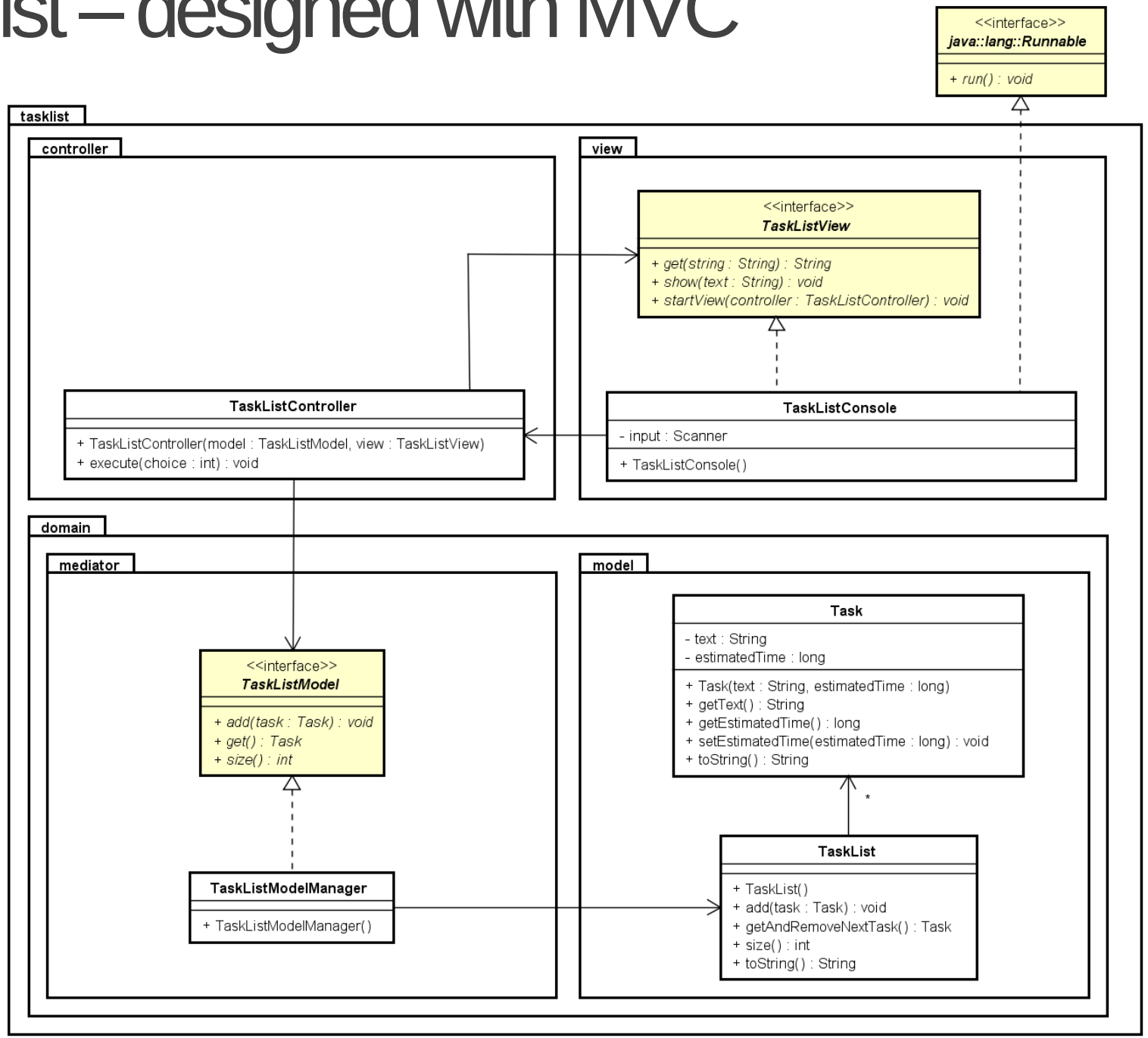
Task List – Main (2/3)

```
switch (choice)
{
   case 1:
      String what = "task";
      System.out.print("Enter " + what + ": ");

      String taskText = input.nextLine();
      what = "estimated task time";
      System.out.print("Enter " + what + ": ");
      String taskTime = input.nextLine();
      long time = -1;
      try {time = Long.parseLong(taskTime);}
      catch (NumberFormatException e) {}

      Task task = new Task(taskText, time);
      taskList.add(task);
      System.out.println("ADDED: " + task);
      break;
```

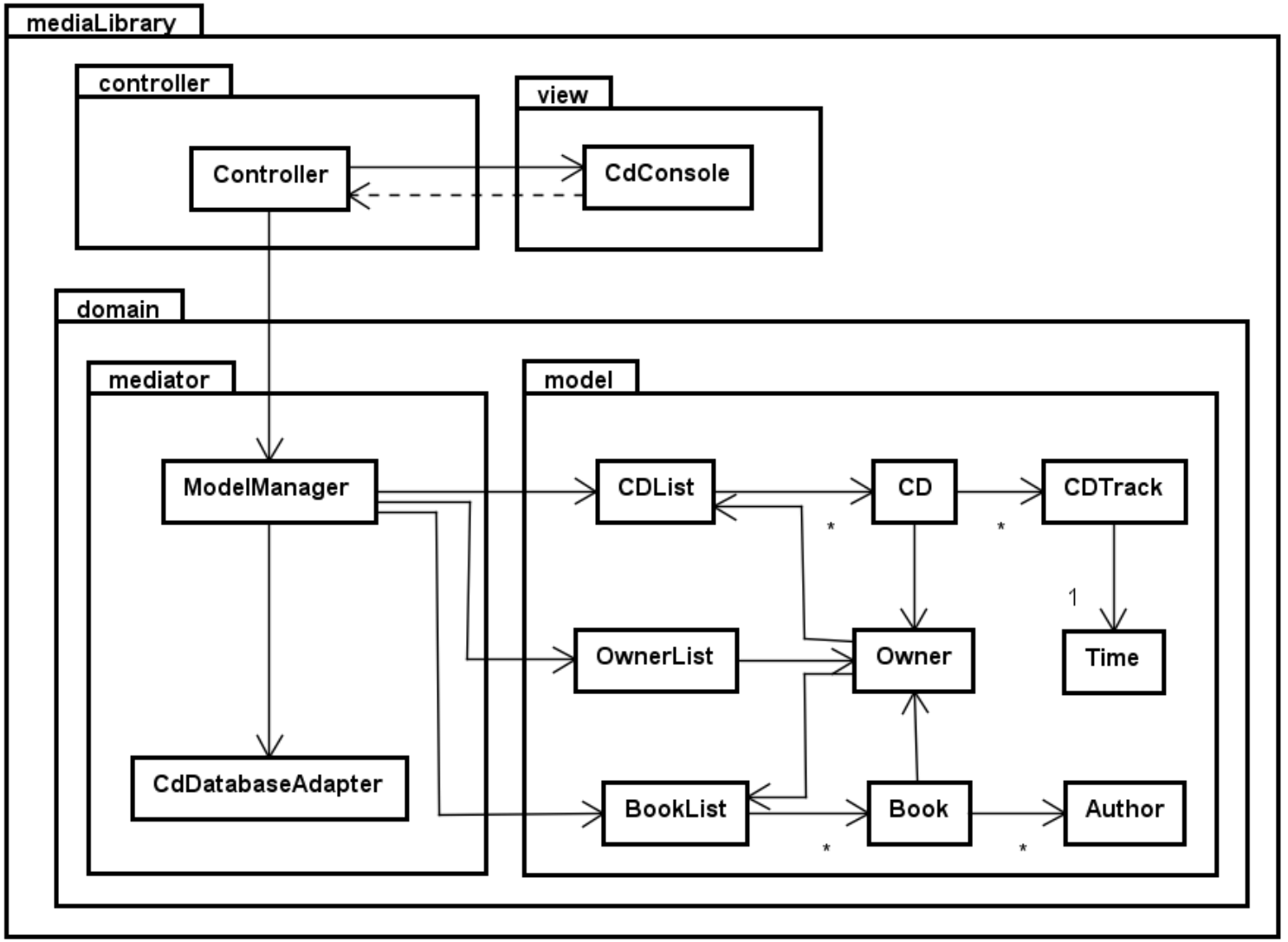
Task List – Main (3/3)

```
            case 2:
               task = taskList.getAndRemoveNextTask();
               System.out.println("Task: " + task);
               break;
            case 3:
               int size = taskList.size();
               System.out.println("Size=" + size);
               break;
            case 0:
               System.exit(1);
               break;
         }
         if (choice == 0)
         {
            continueWorking = false;
         }
      }
      input.close();
   }
```

Task list – designed with MVC



Example: A larger model

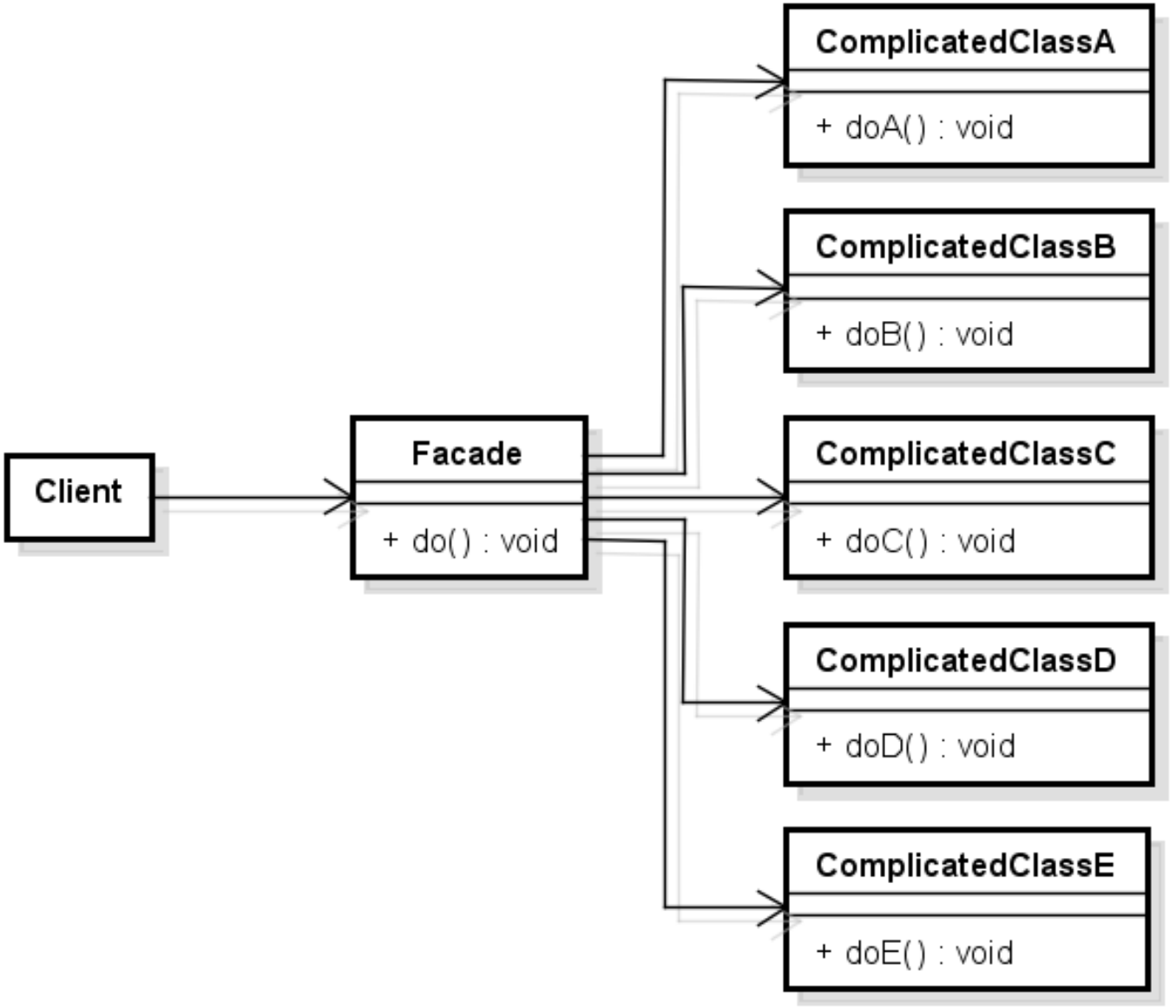


Design Pattern: Façade

- Intent
  - Facade defines a higher-level interface that makes the subsystem easier to use. Wrap a complicated subsystem with a simpler interface.
- Example
  - A model manager as a facade to the model.
- Problem
  - A segment of the client community needs a simplified interface to the overall functionality of a complex subsystem.

Ref: http://sourcemaking.com/design_patterns/facade

Façade design pattern



ModelManager as a Façade (you'll get it for free)