

Problem Set 5

Due: 11AM, Friday October 21, 2005

TIVO 3 – GUI for Recordings [100 points]

Introduction

Most modern programs that interact with humans may be termed GUIs – graphical user interfaces – rather than text-based console applications. Problem sets 5 and 6 will build on problem set 4 by having you create a GUI to handle the selection of television programs to be recorded. Figure 1 below shows a suggestion for the look and feel of the visual interface.

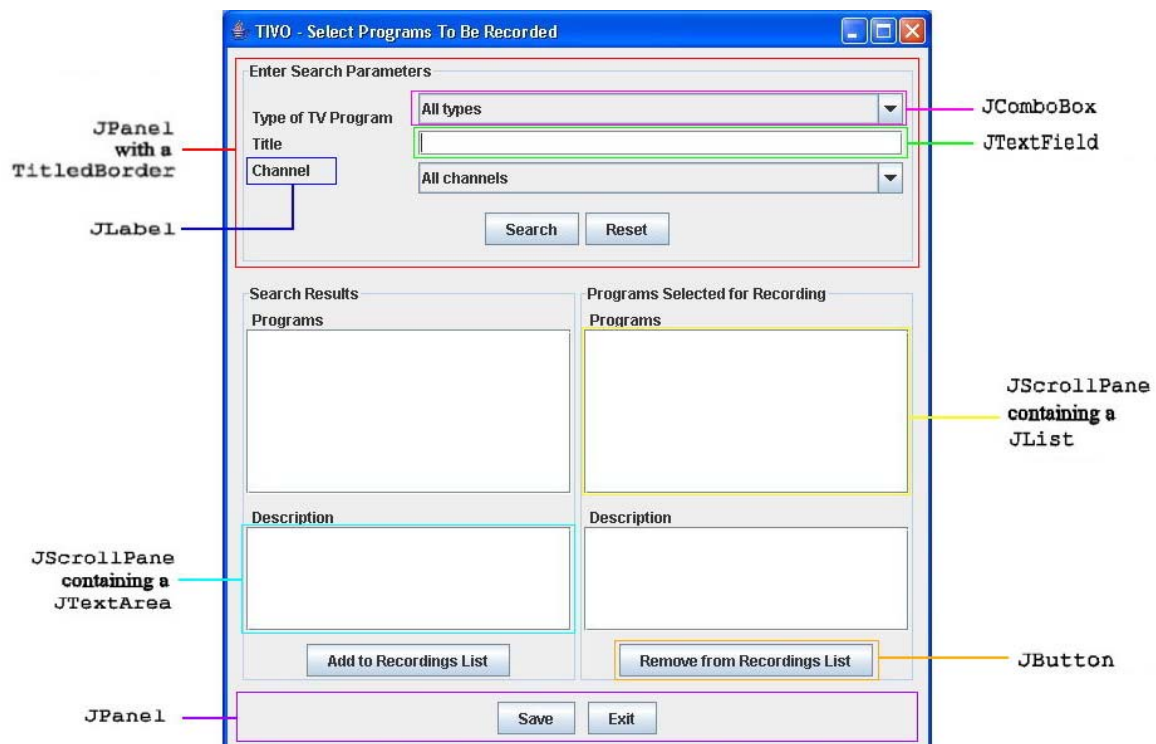


Figure 1: An example of the GUI for television program recording. Key features are highlighted and the classes used to create them are noted.

To build the GUI properly, you should understand how it will work. The input fields (JComboBox and JTextField) are completed as necessary by the user. Pressing the

“Search” button (`JButton`) initiates a search of `DailySchedule` objects and displays the results (`TVPrograms`) in the “Programs” list (`JList`) within the “Search Results” panel (`JPanel`). The “Reset” button causes the textfield to become blank and changes each combo-box to return to its default value (“All types”/“All channels”).

Once the “Search Results” list is populated, if the user selects an item in the list, detailed information on the selected program is displayed in the “Description” area (`JTextArea`). Clicking the “Add to Recordings List” button while a program is selected causes that program to be added to the “Programs Selected for Recording” list.

Similarly, selecting an item in the “Programs Selected for Recording” list will cause the program’s information to be displayed in the “Description” area below the list. Clicking the “Remove from Recordings List” button while a program is selected causes the program to be removed from the “Programs Selected for Recording” list.

You will need the code completed in problem set 4. You may use your own code or the solutions but the code must function as specified.

In problem set 5, you will design the graphical components (the “view”) and write code that “computes” and returns information (the “model”). In problem set 6, you will use the “computation” code to give functionality to the visual components (the “Controller”).

Assignment

Create or modify the following Java classes, as noted in the detailed list below:

`TVProgram.java`, `News.java`, `Movie.java`, `Special.java`, `Series.java`, `RecordingView.java`.

1. `TVProgram.java`

- Declare an abstract method `generateFormattedInformation()`. This method returns a `String` which contains information about the `TVProgram`, formatted appropriately for the “Description” text area.

2. `News.java`

- Implement the `generateFormattedInformation()` method. The resulting `String` should contain:
 - i. The title
 - ii. The hosts
 - iii. The type(s) of the program
 - iv. The description
- Write a static method `allTypesToStringArray()` which returns an array of `Strings` representing all the possible types of a `News` object. For example, a `News` object may be classified as “News”, “Local”, “National”, “International”, “Politics”, etc.

3. `Movie.java`, `Special.java`

- Implement the `generateFormattedInformation()` method. The resulting `String` should contain:
 - i. The title

- ii. The actors
- iii. The type(s) of the program
- iv. The description
- v. The date originally released
- vi. The rating
- Write a static method `allTypesToStringArray()` which returns an array of `Strings` representing all the possible types of a `Movie` or `Special` object.

4. `Series.java`

- Implement the `generateFormattedInformation()` method. The resulting `String` should contain:
 - i. The title
 - ii. The actors
 - iii. The type(s) of the program
 - iv. The description
 - v. The date originally released
 - vi. The season and episode
 - vii. The rating
- Write a static method `allTypesToStringArray()` which returns an array of `Strings` representing all the possible types of a `Series` object.

5. `RecordingView.java` ["view"] (Note: This is a new class)

- Implement the graphical components of the GUI. You may use a layout similar to that suggested earlier (Figure 1). Your GUI should have:
 - i. Text inputs for type, title and channel. The defaults for these input fields should be "All types", "" (empty `String`) and "All channels", respectively.
 - ii. Search and Reset buttons.
 - iii. A list of the programs returned by any search.
 - iv. A list of programs selected for recording
 - v. Text areas to display information on programs selected in either list above.
 - vi. Save and Exit buttons.
- Make sure to have a "model" data member. In this implementation, the "model" will be an object of the `TIVO` class.

6. `TIVO.java` ["model"]

- Write a static `getTVProgramTypes()` method. This method takes no arguments and should return a `String` array containing all the possible types of `TVPrograms`. Note that "TVProgram" itself should not to be included among the possible types.
- Implement a `find()` method which takes in three `Strings` and returns a `Vector` of `TVPrograms`. The `String` inputs represent type, title and channel (the inputs in the GUI). "All types", "" (empty `String`) and "All channels" are the defaults for the text inputs in the GUI.

7. `ProblemSet5.java`

- Test your code. As in problem set 3, a `getDailySchedules()` method has been provided to generate test data.

- i. Create an instance of the `TIVO` class with one `DailySchedule` array.
 1. Print the schedule.
 2. Search for and print all of the “News” programs and all the “Comedy” programs.
- ii. Create an instance of `RecordingView` and make it visible. Note that the GUI will not be functional. Just make sure that all the elements display correctly.

Turn In

- Turn in **electronic** copies of **all source code** (`.java` files). No printed copies are required.
- Place a **comment** with your full name, MIT server username, tutorial section, TA's name, and assignment number at the beginning of all `.java` files in your solution.
- Remember to **comment your code**. Points will be taken off for insufficient comments.
- Place all of the files in your solution into a **single zip file**. Submit this single zip file under the appropriate section and problem set number.
- Your solution is **due at 11AM**. Your uploaded files should have a time stamp of no later than 11AM on the due date.
- **Do not** turn in compiled byte code (`.class` files) or backup source code (`.java~` files).

Penalties

- **30% off** If you turn in your problem set **after 11AM on Friday** but **before 11AM on the following Monday**.
- **No Credit** If you turn in your problem set **after 11AM on the following Monday**.