Lab 7 – Model, View, Controller (MVC) Design Pattern

Computer Science 2334

Due by: Friday, April 8, 2011, 11:00 am

Members: _____

_____

_____

_____

_____

_____

Objectives:

1. To learn how to create a model in the MVC design pattern.
2. To learn how to create a view in the MVC design pattern.
3. To learn how to create a controller in the MVC design pattern.
4. To learn how the model, view, and controller are connected to one another and communicate with one another in the MVC design pattern.
5. To demonstrate this knowledge by completing a series of exercises.

Instructions:

This lab exercise requires a laptop with an Internet connection. Once you have completed the exercises in this document, your team will submit it for grading.

Make sure you read this lab description and look at all of the source code posted on the class website for this lab exercise before you begin working.

Assignment:

Graphical User Interfaces using the Model, View, Controller (MVC) design pattern are an important programming abstraction that shows how additional structure can be built from objects; it is also one that will be used in future projects. Carefully inspect how it works and the documentation comments included in the code.

1. Download the `Lab7-eclipse.zip` project archive from the class website. Import the project into your Eclipse workspace using the slides from Lab 2. You will submit the modified project archive when you are finished.

2. Review the source code posted on the class website. Pay close attention to the MVC classes, which are the **Model**, **StateView**, **PlaceView**, and **Controller** classes.

3. Read through the source code of the **Model** class and note the comments provided in the source code that give hints as to what needs to be done in the program.

4. Declare variables called `statePlaceList` and `selectedPlaceList` to hold the information in the model for the application objects being modeled. These variables must be of appropriate types to hold lists of **Place** information. (See the **Place** class for details on what information is available for each place.)

What data structures will be used for `statePlaceList` and `selectedPlaceList` and why?

5. Declare a variable called `actionListenerList` to hold the information in the model that allows it to be the source for action events. This variable must be of an appropriate type to hold a list of listeners.

What data structure will be used for `actionListenerList` and why?

6. Complete the **model** class by adding a call to its `processEvent()` method. This method should be called with an **ActionEvent** that has "Select Place" as its command string.

Where should `processEvent()` be called? Why?

7. Read through the source code of the **StateView** class and note the comments provided in the source code that give hints as to what needs to be done in the program.

8. Complete the `setModel()` method for **StateView** by registering the view.

Will the **StateView** object be a source or a listener in this part of the code? Why?

To which class will the object at the other end of this source/listener connection belong? Why? Will that other object be a source or a listener? Why?

9. Read through the source code of the **PlaceView** class and note the comments provided in the source code that give hints as to what needs to be done in the program.

10. Complete the `setModel()` method for **PlaceView** by registering the view.

Will the **PlaceView** object be a source or a listener in this part of the code? Why?

To which class will the object at the other end of this source/listener connection belong? Why? Will that other object be a source or a listener? Why?

11. Read through the source code of the **Controller** class and note the comments provided in the source code that give hints as to what needs to be done in the program.

12. Complete the `setView()` method for the controller by registering the controller.

Will the **Controller** object be a source or a listener in this part of the code? Why?

To which class will the object at the other end of this source/listener connection belong? Why? Will that other object be a source or a listener? Why?

13. Test the program to make sure it correctly responds to list selection events.

14. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary. (If you can't figure out how to fix your code to avoid the cast warning on the cloned `actionListenerList`, you may leave in that warning.)

15. Submit the **project archive** following the steps given in the **Submission Instructions** by **Friday, April 8, 11:00 am** through D2L (http://learn.ou.edu).

16. Turn in this lab handout (with completed answers) to your lab instructor during lab hours or by bringing it to Professor Hougen's office by **Friday, April 8, 11:00 am** and handing it to him or sliding it under his office door if he is not there.