WESTERN UNIVERSITY - CANADA
FACULTY OF ENGINEERING,
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SE 3352A – SOFTWARE REQUIREMENTS AND ANALYSIS

Laboratory 4: Mediator Design Patterns

Due Date: November 21, 2019

1 Why?

In software engineering, a software design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations.

In this lab, we will study and practice Mediator design pattern. Mediator pattern defines an object that encapsulates how a set of objects interact, promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.

2 Learning Objectives

- Understand how to apply Mediator design pattern techniques to certain problems.
- Gain hand-on practice on using Mediator pattern, learning how to reduce communication complexity between multiple GUI objects and make them reusable objects within JavaFX environment.

3 Success Criteria

- Be able to understand the essence of the Mediator design pattern.
- Be able to apply Mediator design pattern to a design problem.

4 Resources

- The SE3352a class attendance,
- Lecture notes Unit4 slides 149-160, and the Reading below.
- The attached files; countries.txt, province-states.txt, cities.txt, and derby.jar.

5 Plan

1. Study the Reading section below.
2. Review the Mediator design pattern given in Unit 4 slides 149-160.
3. **Individually**, answer the Key Questions, do the Exercise, and solve the Problems.
4. Open a new Word document and name it "yourUWOid_lab4.docx". For example, if your UWO email is *aouda@uwo.ca* then the file name will be "*aouda*_lab4.docx"

5. Write all your answers including all your drawing (if any) in this Word file.
6. Using NetBeans IDE or IntelliJ Idea IDE export your Java project into zip file called yourUWOid_Lab4-Java.zip.
7. Archive your yourUWOid_Lab4-Java.zip along with your yourUWOid_lab4.docx file and name it "yourUserID_SE3352a_lab4.zip"
8. Submit this zip file "yourUserID_SE3352a_lab4.zip" through OWL for grading. Note that, any section decorated (Deliverable) is the section that you need to complete and submit to the instructor.

6 Reading

The class diagram shown in Figure 1 represents a client/server based messaging application. This diagram uses a mediator design pattern that facilitating loosely coupled communication between different participants registering with a class forum. In this diagram, we have one type of participant (Students). The ClassForum is the central hub through which all communication takes place. The MainDriver is just a demo to show and test the objects creations and their communications. In this simple class, one object of type SE3352ClassForum is created called "SE3352a" and two objects of type Students are created as well and are called "George" and Paul", these two students are then registered into the SE3352a object. To test the objects communications, this demo class is sending a "Hello" message from "George" to "Paul".

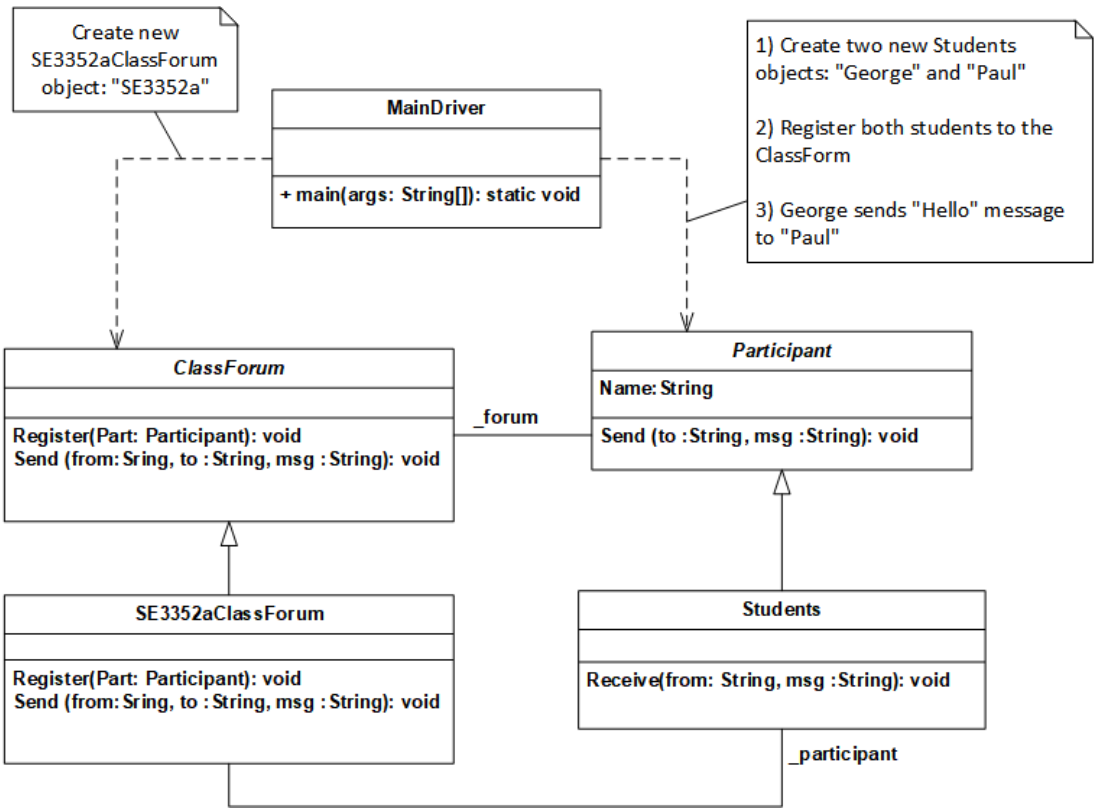

Figure 1

7 Key Questions (Deliverable – out of 4 marks)

1. When to use Mediator design pattern?
2. What are the Consequences of using Mediator design pattern?

8 Exercises (Deliverable – out of 4 marks)

Study the class diagram shown in Figure 1 and consider the given scenario in the Reading Section. Draw a UML sequence diagram that reflects this scenario showing the creations and the communications among the participating objects.

Notice If you have some questions or if you need to make sure you are doing the right answers, your TAs are willing to help and they are available during the designated Labs hours.

9 Problem (Deliverable – out of 12 marks)

You have decided to develop a JavaFX GUI application that facilitates the employee profiles data entry process. Your application will have multiple FXML forms, such that:

- The main form has a main menu with two menu items; File and Employee Profile. The File menu item has the option to close the application, while the Employee Profile menu item has two sub options; Add Profile to open Add Profile form and Modify Profile to open View/Modify/Delete profile form. See Fig 2a, 2b.
- The Add Profile form, as shown in Fig 2c, has the following UI controls:
  - 4 Textfields for Employee ID, Name, Postal Code, and Street Address.
  - 3 drop-down lists (ComboBox) for Country, Province/State, and City.
  - 2 buttons for Cancel and Save.
- When the Add Profile form starts, Employee ID, Name, Country, and Cancel controls are enabled and all other controls are disabled. User starts by entering the employee ID, name, and then the country field. When the user click the country field it displays a list of country names. When the user selects one country, the Province/State control will be enabled and should display a list of related province/states related to the selected country. When one region is selected, the city control will be enabled and displays a list of the cities located within this selected region. When the user select one city the Postal Code, and Street Address fields in addition to the save button will be enabled. The user then can enter the postal code and the street address information. When save button is clicked the entered information of the employee will be saved. If Cancel button is clicked, the form will be closed without saving any data.
- Mediator pattern should be used in order to control the execution of the three ComboBoxs and the two buttons as described above. Your implementation should be such that these controls (ComboBoxs and buttons) are reusable controls, i.e., When you implement a ComboBox for the country field, you can reuse it (copy-and-paste) for the province field and also for the city field without changing anything of its implementation within the form except the control name. Note that, when these controls have a change (like country selected) they declare (fire an event) that they have an update and it is up to the mediator class to determine which control raised the event and what to do accordingly.

- The data needed for the countries, regions, and cities are given in three separate text files, you need to use them all. Feel free to choose any mechanism to maintain the data of this application, you can save then temporary in the memory with any data structure or save then into a database using DerbyDB. Using a database is recommended for this application.
- The View/Modify/Delete Profile form, as shown in Fig 2d, has the following UI controls:
  - 3 Textfields for Name, Postal Code, and Street Address.
  - 4 drop-down lists (ComboBox) for Employee ID, Country, Province/State, and City.
  - 3 buttons for Cancel, Save, and Delete.
- When the View/Modify/Delete Profile form starts, Employee ID is enabled, and all other controls are disabled. User starts by selecting the employee ID from the drop-down list of the saved employee profiles, then the rest of the fields get populated and become enabled for any changes. The functionality of these fields is the same as described above. Note that, the ComboBoxes and the Buttons can be copied and pasted from the Add profile form without the need to change anything. You need to add a Delete button that will enabled only when the employee data is shown. When Delete button is clicked, the employee profile data is deleted, and the form is closed.
- Again the Mediator pattern should be used to control the execution of the controls of this form.

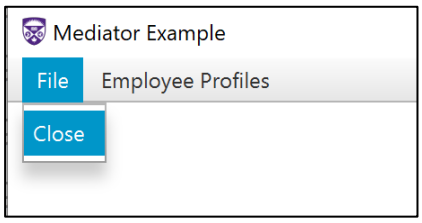

Figure 2a

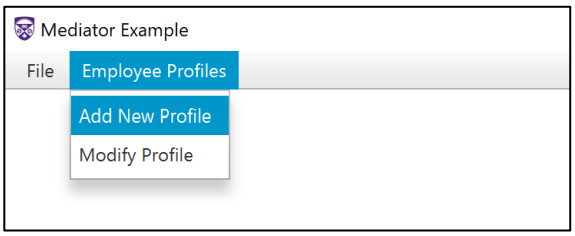

Figure 2b

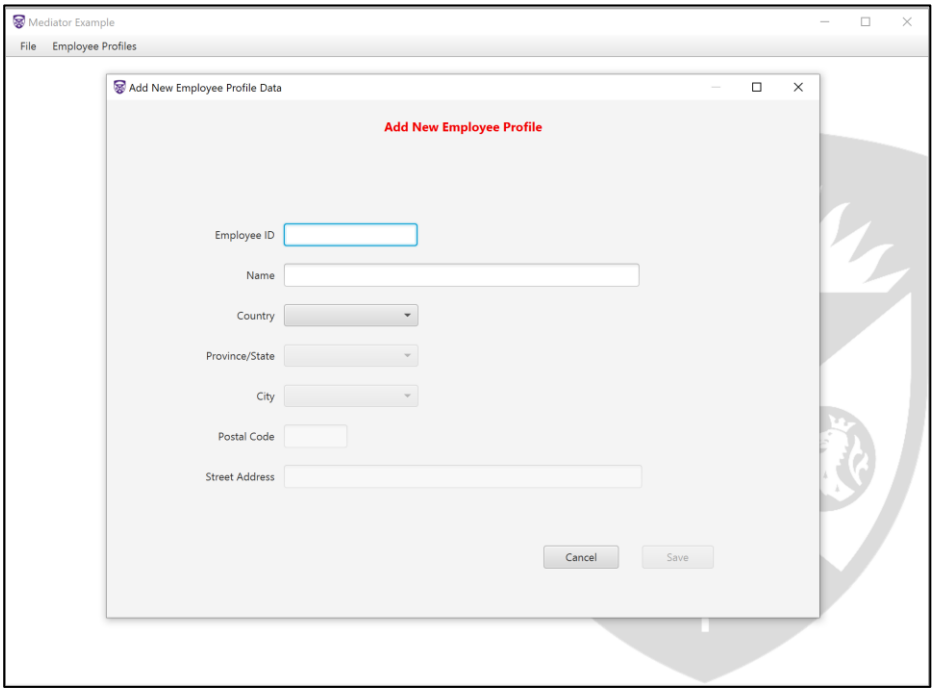

Figure 2c

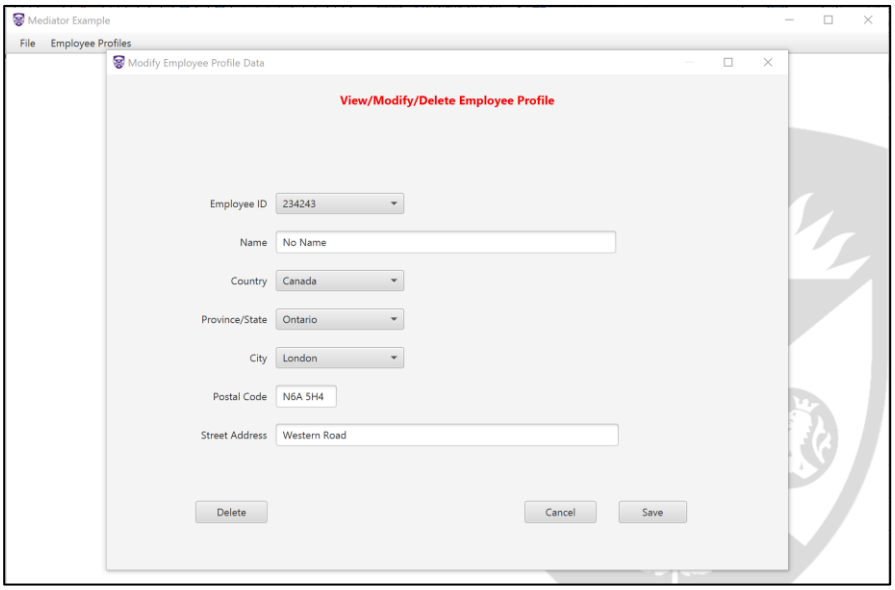

Figure 2d