DPIT121 - Object Oriented Design and Programming

Lab 7 (due in week 10 Lab)

The purpose of this lab is to practice building a desktop GUI based program in Java. You should be able to use all the logical classes from your assignment II without any changes. You even can start from assignment I sample solution. The only class that you need to change is your text based UI which should be changed to Swing based GUI (several JFrames). You may use Swing Toolbox. This will be the first iteration of GUI design for your Assignment III

A jar file of a mobile company software (java executable program) is provided with this lab which is the demo of a similar application to the one you need to implement. Run the demo to see exactly what you need to do. Also watch lab 7 tutorial (located in "Recorded Lectures" folder as well as lab 7 folder) to understand all the elements of the demo. Also study the BankUI lecture code to understand how to do this lab. In this lab we only consider User Login. Admin login will be done in your assignment III.

The program starts with a login page. You can log in with "john123" and "password1". You can click on the "list of users" menu to see a list of all the users and passwords. The menu also has options to save insuranceCompany and load it from file. A sample database binary file should be provided with the program (the same as the demo). Test code includes several users but you can add more policies or change the user information and save it.

Note: Login JFrame has a field InsuranceCompany insuranceCompany and this is the relation between your UI and the Logic/Domain of your software.

After successful login the UserUI form will be displayed with several tabs. Note that UserUI JFrame has several fields that will be send to its constructer from Login form: ✓

ArrayList<String> cities; //a list of distinct user cities that is generated from a corresponding method populateDistinctCities() inside insuranceCompany. ✓
User user; // the user that has been validated in the login process and sent to UserUI ✓
Login login; // a pointer to the login form to be able to return after logout. ✓

Work with each tab left to right and implement it in your code:

1) Edit Personal Data: To edit the current user information. Check the demo to see that a list of distinct city names in the company is populated in the comboBox. You can add a new city by ticking the checkbox and type the new city name. The new city will be added to the comboBox as well. Wrong data type (e.g., string instead ✓

of integer for street number) will throw an exception that is caught in a catch block which displays a Message Box.

2) Add Policies: To add a policy to the user. Check the demo to see that how choosing the proper radio button will change the text fields. Any errors result in a Message Box. The policy will be created and sent to user.addPolicy() method. You need to have radio buttons for ThirdParty and Comprehensive policies and update the data fields accordingly. ✓

3) Find a Policy: To search for a policy and display it in a text area. If ID is not found a proper message will be displayed in the text area. Note that the policies are appended to the Text Area. ✓

4) Policies Information: To display all the user policies in a Table. See the demo that how different mobile plan types in the demo include NA (Not Available) fields (e.g., Personal Plan has NA for ABN and Number of Employees). You should do exactly the same thing for ThirdParty and Comprehensive policies. There are buttons to Delete and Update the selected policy in the table (when you click and select it the whole row will be highlighted). Right click on the selected row and you will see a popup menu will be appeared which can be used to do the same tasks. Try to also implement the popup menu as well.
Clicking on update button will open a new JFrame (UpdateUI) . This will be explained later. ✓

5) Filter Policy: To initially display all the policies. Typing in the Car Model text field will filter the list of policies based on car model automatically (by calling InsurancePolicy.filterByCarModel()). In addition you can type a date as YYYY/MM/DD in the Expiry Date field and click on "Filter by Date" to filter the list based on the date in addition to the car model. If fields are empty, no filtering based on either model or date will be applied. Note that the demo has the same scenario to filter a list of plans based on mobile model and plan expiry date.
In addition, there is a checkbox "Sort by Name" which can be used to show the list of policies based on original order or sort them by name (using Comparable and Collections.sort() or sortPolicies() method inside the user). The algorithm to do this is as below:

Anytime the panel is loaded or user types a character in the car model textfield or clicks on the filter by date, or ticks or unticks the checkbox to sort by username (Note: these are all events that can be redirected) do this:

- Create an ArrayList<InsurancePolicy> policies as a shallow copy of all user policies.
- if modelText is not empty then policies= InsurancePolicy.filterByCarModel(policies, model).
- if expiryDateText is no empty split it by "/" and create a MyDate object and then call the filterByExpiryDate the same as the other one to filter the list again.
- if the sort CheckBox is selected then sort the policies by using policies=Collections.sort(policies)
- fill the table by iterating over policies ArrayList.

The UpdateForm has these fields:

InsurancePolicy policy;// the policy to be populated and updated

UserUI userUI;// a pointer to the userUI to be able to return to it and also update the tables

When user clicks on update button in policies information Tab (or using the update option from the popup menu) the UpdateUI form will be displayed. The policy details will be populated in the form. User can change whatever and click on update to update the record. Note that how the policy type is detected (by using instanceof). However the type as well as ID is read-only and cannot be changed.
Note that clicking on logout menu in the UserUI will redirect the program to login form again. User can choose the save menu to save all the changes in the file.

You need to implement all three forms and all tabs and your program should work exactly the same the demo.

Marking Guide:

1- Login Form is done and working with different users. Wrong login will display an error message: 0.5 marks
2- Each Tab in UserUI form is done exactly the same as the demo: 0.75 per Tab and totally 3.75 marks
3- UpdateUI form is done exactly the same as the demo: 0.75 marks

Total Lab mark: 5 marks