**Flight Reservation System**

Design

The application makes use of two design patterns – the Singleton and Observer pattern. Singletons are used to ensure only one instance of a class is created, and to provide a global access to it. In the application, Singletons are used for the controllers – controllers hold the lists of objects and should be instantianted only once to ensure only one list is getting manipulated.

The Java built-in Observer pattern is used to automatically update content in the view after it has been changed in the models. After the view and controllers have been created, the view is added as an observer to the controllers. All controllers extend from an AbstractController which extends from the Observable class. When the data in the models is changed, setChanged() is called to indicate the Observable object has changed and notifyObservers() is called which leads to an update of the data in the view.

The other purpose of the AbstractController is to hold the serialization and deserialization methods that all concrete controllers have access to as they inherit from AbstractController. As all the Serializable classes implement the interface Model, generic serialization and deserialization methods could be called which increases code reuse (two instead of writing eight different methods in total). With that in mind, the same data structure - ArrayList – was used for storing all collection data, so that they would be compatible with the data structure used in the shared serialization/ deserialization methods.

Polymorphic Programming

Polymorphic programming is used where a passenger’s class needs to be determined – the list of passengers is searched without knowledge of if the passenger who will be passed into the method is a standard or business class customer. The class is determined at runtime, when the method getPassengerClass() is called. The method is implemented in the correct sub-class, returning the enumerator value for class.

Enumerators

I have used enumerators on two occasions – to be used as indicators of standard or business class, and to list possible destination and departure values. Using enumerators instead of strings reduces the risk of errors due to mistyping values. In the program, numerators were used as both the indicators of passenger class and flight locations can only be assigned certain values out of a set of pre-determined list.

GUI

The GUI consists of a single JFrame with additional popup dialogs for user input, information or error messages. As the system starts up, the user is given a choice to load data or start with an empty system. The card layout is used to switch content of the GUI based on which section of the reservation the user is in, i.e. flight, airline or passenger manager. A table model is used to display a list of flights. This allows for the user to select a row and upon clicking “view passengers” the passengers for the chosen flight are displayed. Buttons with ActionListeners are used to call on methods in the controller or to launch popups.

New techniques

The application makes use of the model-view-controller paradigm to ensure a separation of concerns. The model classes represent the data, the view is made up of the user interface, and the controllers contain methods that use and manipulate the data in the models. The purpose of using the MVC pattern is to ensure higher levels of decoupling and code reuse. For example, this is done by ensuring that the view is only responsible for displaying information that it has received from the controller, without any underlying ‘knowledge’ of how the data is stored or retrieved. If the structure of models needs to be changed, this will then not affect the display methods and thus does not break the code in the view.

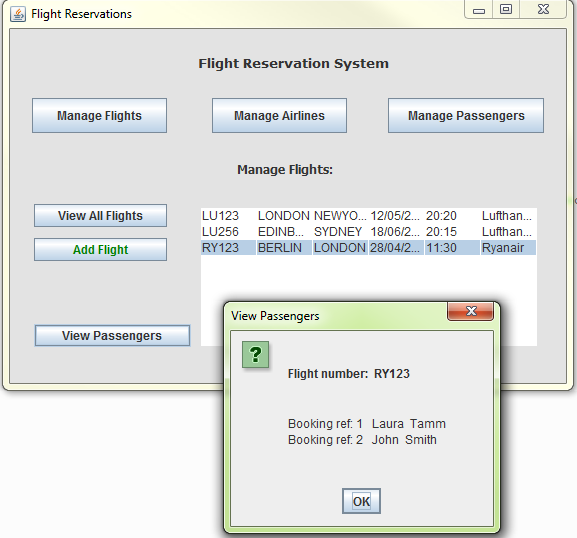


Figure : View Passengers

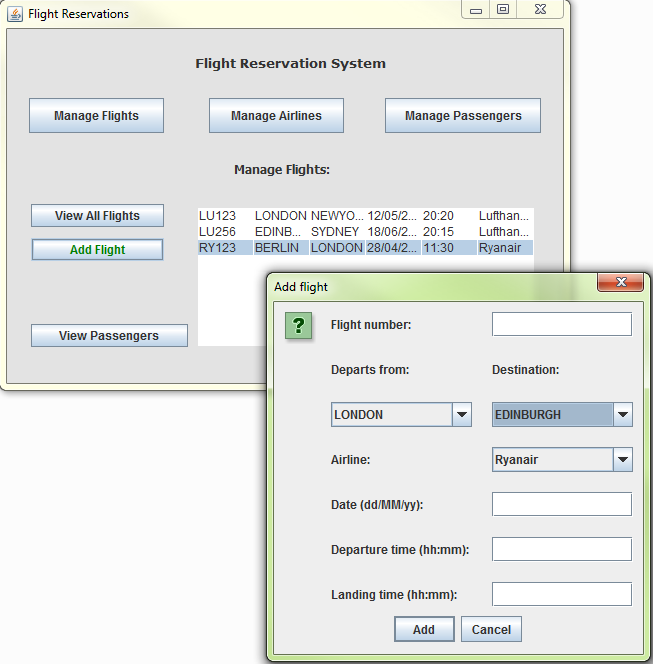


Figure : Add flight

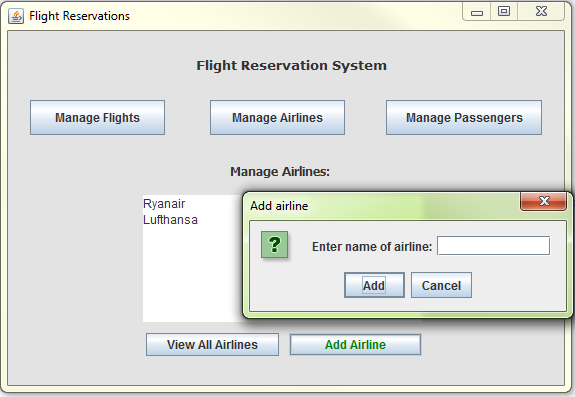


Figure : Add airline

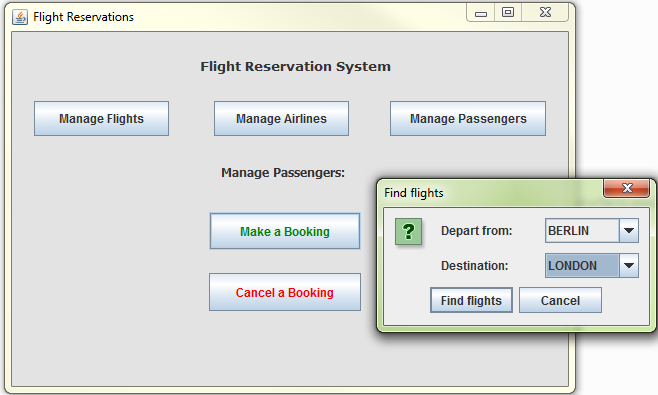


Figure : Book flight - step 1 find flights for route

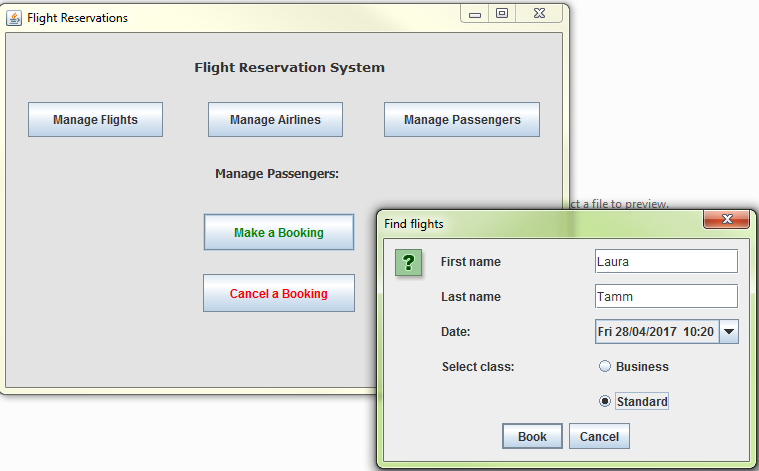


Figure : Book flight - step 2 continue booking

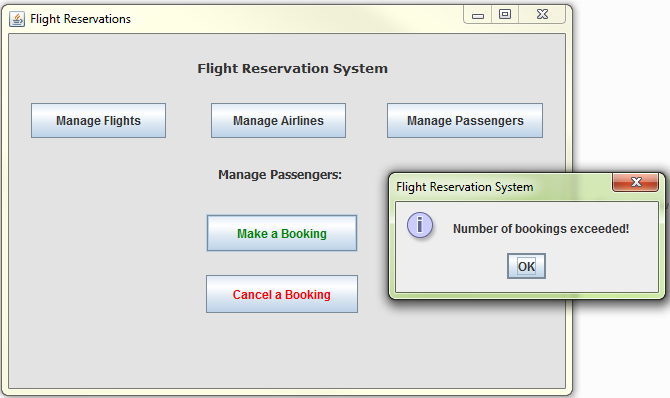


Figure : Error if booking allowance exceeded