The University of the West Indies, St. Augustine
COMP 2603 Object Oriented Programming 1
Assignment 3
2017/2018 Semester 2

Due Date: April 18, 2018 at 11:50 p.m.

Submission Instructions
The code for each class in the application should be written in separate source files. Ensure your student ID is documented in each file. Upload a zipped file of your source to the myElearning course page by the deadline.

Overview:
An object-oriented application is required to manage the passenger records and boarding passes issued by Kaiteki Airlines. The application must allow the following operations to be performed:

- Add a new passenger to the system
- Update an existing passenger's details
- Issue a boarding pass to a passenger for a particular flight
- Change a passenger's seat once a boarding pass is issued.
- Retrieve and display the details of a passenger's boarding pass.
- Retrieve and display a passenger's details
- Clear the data from the textfields and text areas, and reset the GUI

Screenshot of the Graphical User Interface

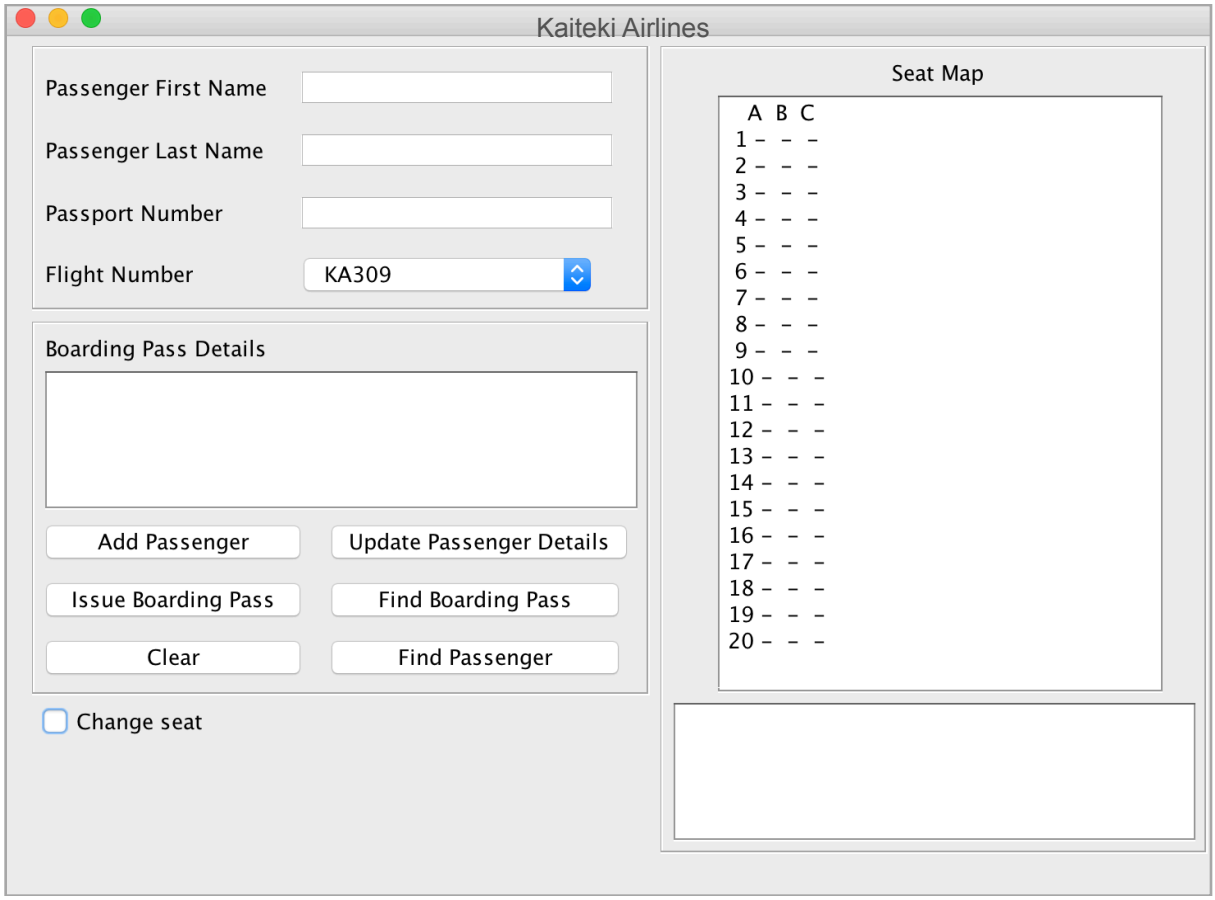

Figure 1. Main GUI upon loading the application

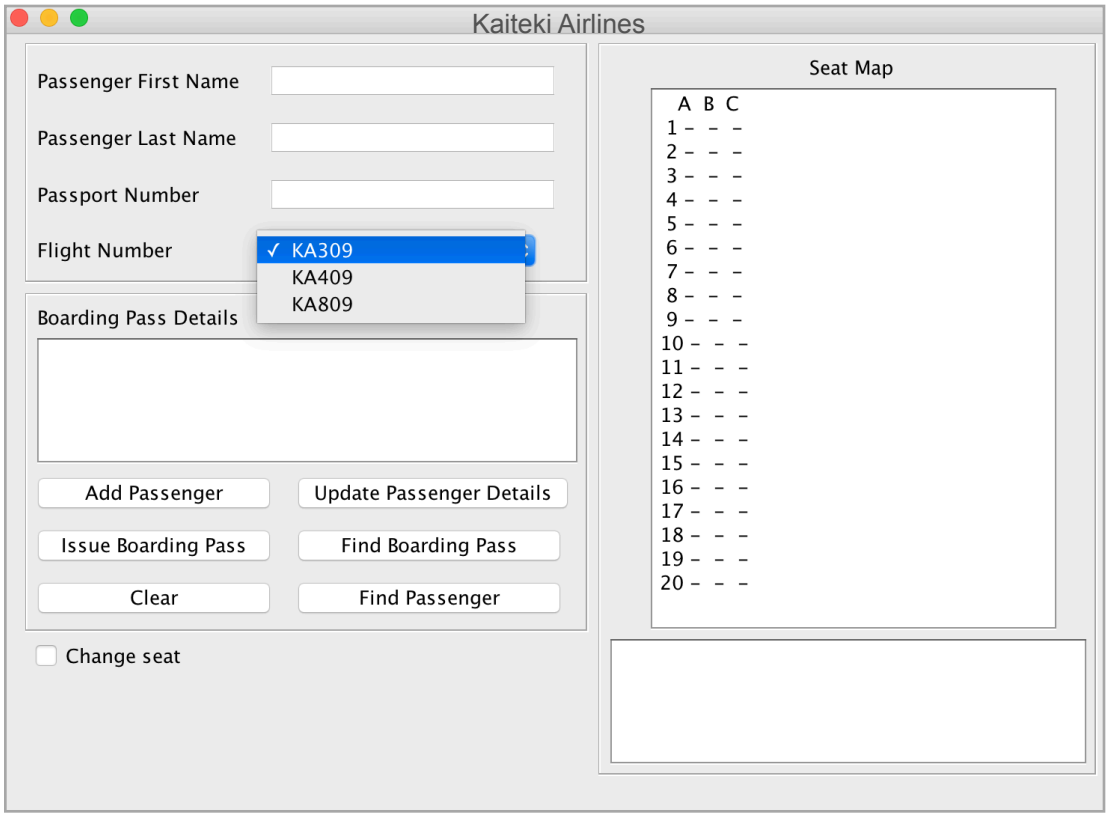

Figure 2. The Change Seat Panel (Displayed when the Change Seat Check box is selected)

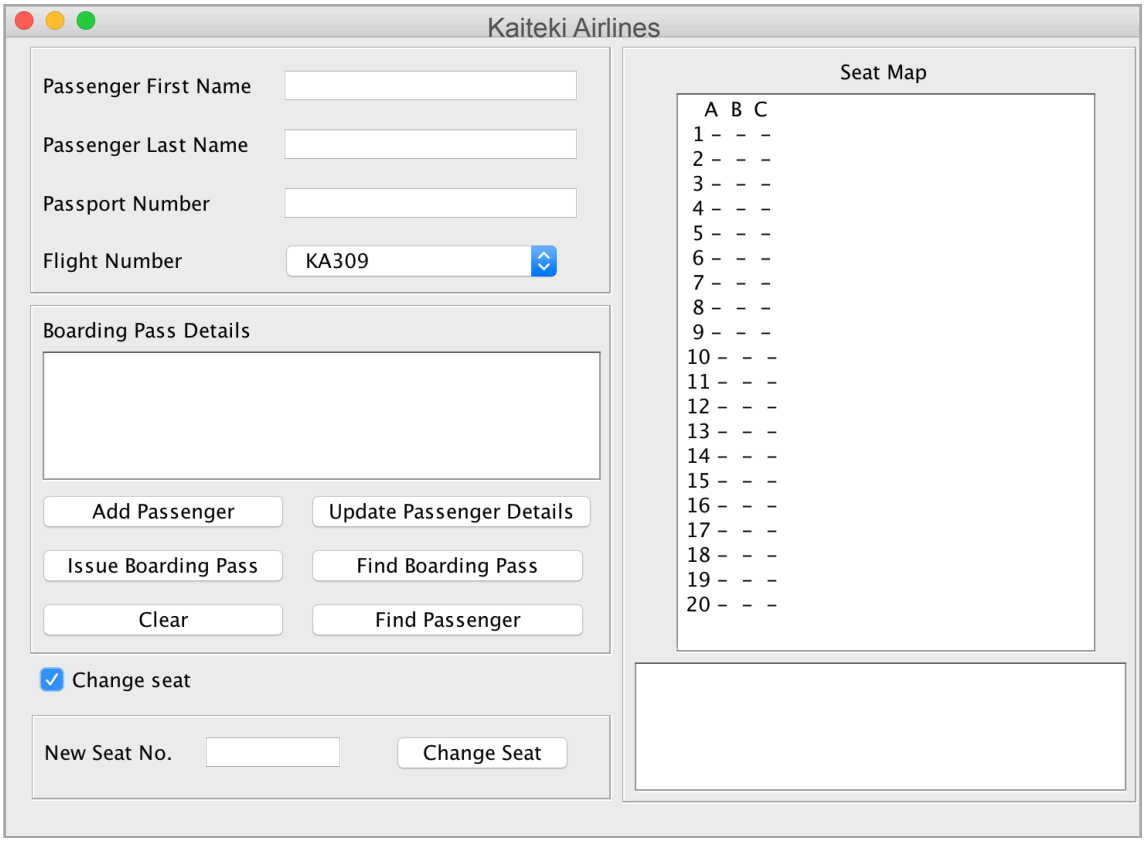

Figure 3. The Flight Number Drop Down List - Sorted Alphabetically

Figure 1 shows the main screen of the Kaiteki Airline GUI. Figures 2 and 3 show the component functionality of the two interactive GUI elements. Figure 4 shows an example of a successful seat change.

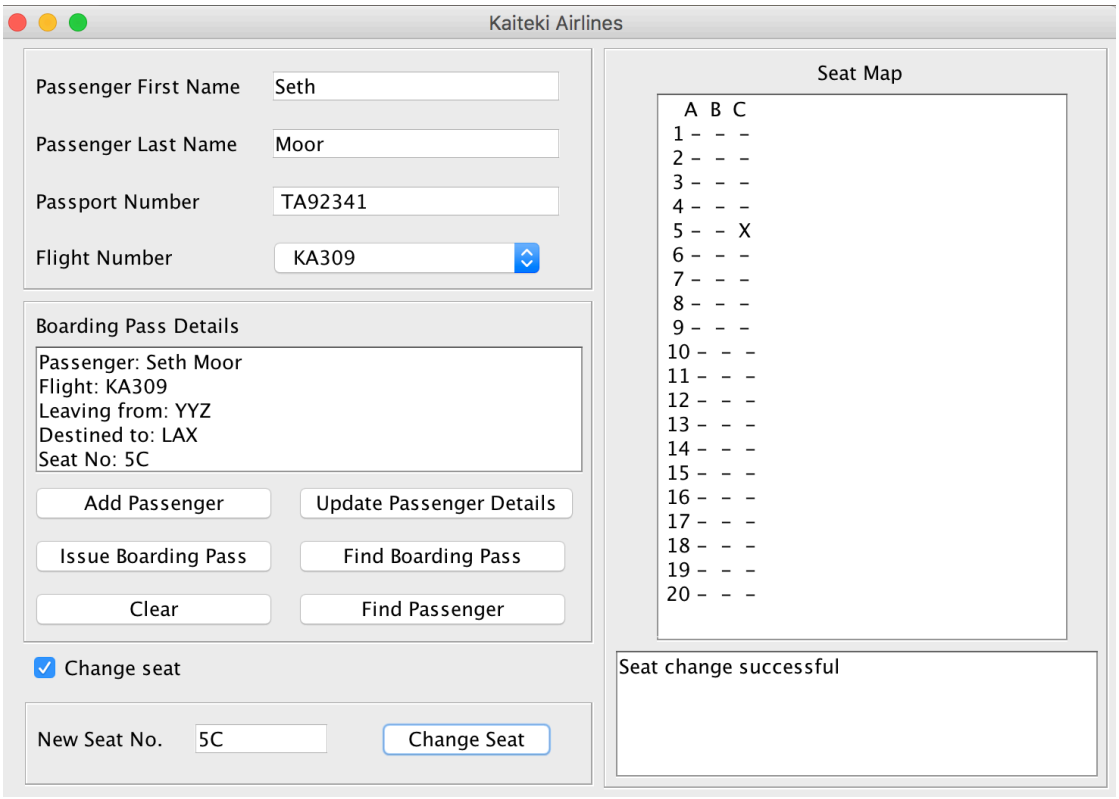

Figure 4. An example of a successful seat change for a passenger on a given flight.

View and Model Classes

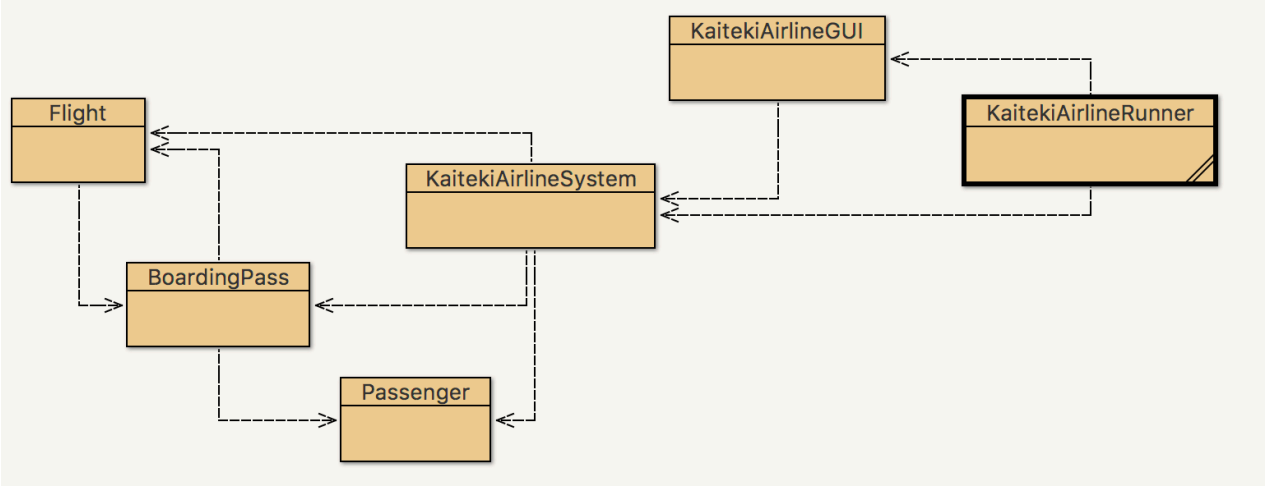

Figure 5. Class Diagram of View Class (KaitekiAirlinesGUI), Model Classes and Runner Class

The application follows the Model-View separation design where the presentation aspects of the application are separated from the processing and domain class functionality. The main classes in the application are shown in Figure 5. The Runner class creates instances of the view and one model class, and associates them.

KaitekiAirlineGUI

The KaitekiAirlineGUI class presents the GUI shown in Figures 1, 2 and 3. Appropriate functionality is required as follows:

- The GUI has the title “Kaiteki Airlines”
- The Flight Number combo box is loaded dynamically with the flight numbers provided by the KaitekiAirlineSystem class as shown in Figure 2. The first flight should be selected by default.
- When the Change Seat check box is selected, the GUI elements in Figure 3 should be displayed.
- When the application first loads, the Change Seat check box is unselected, the seat map for the first flight in the combo box should be displayed as in Figure 1, and all text fields and text areas should be cleared of data.
- Full error handling of invalid data entered in the GUI
- Listeners and code to implement the functionality for the buttons listed in the table below. Appropriate messages are to be displayed for all successful and all failed cases for each button (with the exception of the Clear button)

Button	Functionality
Add Passenger	Adds a new passenger to the system. Required data: first name, last name, passport number, flight number.
Update Passenger Details	Retrieves a passenger’s details using the passport number if the passenger is in the system, and displays the passenger’s first name and last name in the appropriate text fields
Issue Boarding Pass	Creates a boarding pass for a passenger for the selected flight if not issued as yet. Assigns a random seat number for the flight, and displays the boarding pass details in the GUI.
Find Boarding Pass	Retrieves a passenger’s boarding pass for the selected flight and displays the details if a boarding pass was issued.
Find Passenger	Retrieves a passenger’s details using the passport number if the passenger is in the system, and displays the passenger’s first name and last name in the appropriate text fields
Change Seat	Changes a passenger’s seat to the one requested if a boarding pass has been issued for the selected flight and if the seat is unoccupied.
Clear	Resets the GUI to the stage of Figure 1

KaitekiAirlineSystem

The KaitekiAirlineSystem class implements the functionality of the GUI. It has the following attributes and methods. You are required to choose an appropriate collection (Map or Collection) object to implement the attributes.

Attribute	Purpose
passengers	A collection that stores Passenger objects such that given a passport number, the appropriate Passenger object is retrieved efficiently. Duplicate passengers are not allowed.

Attribute	Purpose
flights	A collection that stores Flight objects such that given a flight number, the appropriate Flight object is retrieved efficiently. Duplicate flights are not allowed. Flights should be stored in sorted order, ascending by flight number.
boardingPasses	A collection that stores BoardingPass objects such that given a Passenger object, the appropriate BoardingPass object is retrieved efficiently. Duplicate boarding passes are not allowed.

Method Signature	Purpose
String addPassenger (String firstName, String lastName, String passportNo, String flightNo)	Creates a new Passenger object and adds it to the collection of passengers.
String updatePassenger (String firstName, String lastName, String passportNo, String flightNo)	Retrieves a Passenger object by passport number from the collection of passengers and updates the data of the passenger.
ArrayList<String> findPassenger(String passportNo)	Retrieves a Passenger object by passport number from the collection of passengers and returns the first name and last name of the passenger in an ArrayList
String issueBoardingPass(String passportNo, String flightNo)	Creates a new BoardingPass object for a passenger for a particular flight if the passenger is in the system and assigns a random seat to the passenger.
String getBoardingPass(String passportNo, String flightNo)	Retrieves the BoardingPass object details for a passenger for a particular flight if the passenger is in the system and the boarding pass was issued already.
String[ ] getFlightNos()	Returns a String array containing a sorted list of flight numbers in the system. This is used to initialise the flight number combo box in the GUI class
String getSeatMap(String flightNo)	Returns a string representation of the seat map for a flight (as shown in Figure 4) where occupied seats are indicated by an 'X' and empty seats by a '-'.
String updateSeat(String passportNo, String flightNo, String seatNo)	Updates a passenger's current seat to the desired seat for a given flight. The boarding pass and seat map are also updated to reflect the change as in Figure 4.

Flight Class

The Flight class has the following attributes and method. Add accessor and mutator methods as necessary.

Attribute	Purpose
flightNo	The flight's number (Format: XXYYY where X is an uppercase character A-Z inclusive, and Y is a positive integer 0-9 inclusive.

Attribute	Purpose
destination	The flight's destination (IATA 3-character airport code)
origin	The flight's destination (IATA 3-character airport code)
seats	A single collection that stores Boarding Pass objects for the flight according to the seat map of the flight (as shown in Figure 1). A flight has a maximum of 60 seats. A seat is identified by a seat number consisting of a row number (1 to 20 inclusive) followed by a seat letter (A, B or C) e.g. 1A, 12C using the GUI.

Method Signature	Purpose
String getEmptySeat()	Locates an empty seat on the flight at random, and returns the seat number
boolean releaseSeat(String seatNo, BoardingPass bp)	Releases a seat occupied by a passenger on the flight with the given boarding pass. Returns true if successful and false otherwise (e.g. invalid seat number). The boarding pass is updated accordingly.
boolean assignSeat(BoardingPass bp)	Assigns an empty seat to a passenger with the given boarding pass and updates the boarding pass with the assigned seat number. Returns true if successful and false otherwise.
boolean assignSeat(String seatNo, BoardingPass bp)	Assigns a requested seat to a passenger with the given boarding pass and updates the boarding pass with the requested seat number. Returns true if successful and false otherwise.
String getSeatMap()	Returns a string representation of the seats collection formatted as a seat map for the flight. Occupied seats are indicated by an 'X' and empty seats by a '-'.

Passenger Class

A Passenger has a first name, last name, and a passport number. A passenger may have only one boarding pass per flight, however the passenger may take as many flights as are in the system. Model this class based on the requirements for the assignment and guided by the screens in Figures 1-4.

BoardingPass Class

A Boarding Pass has a seat number, a passenger and a flight. The toString( ) method should return a string containing the details shown in Figure 4 in the boarding pass text area. A boarding pass must be updated whenever the passenger's details are updated or when the seat number is updated. Model this class based on the requirements for the assignment and guided by the screens in Figures 1-4.

General Guidelines

- Test your solution thoroughly for as many cases and test sequences possible based on the operations possible using the GUI
- Cater for invalid input entered in the GUI and write appropriate error handling code
- Pre-load the system with at least 5 passengers.
- The system should have at minimum 3 flights.
- Display appropriate messages for every operation
- Use Netbeans for your assignment.