The University of the West Indies, St. Augustine
COMP 2603 Object Oriented Programming 1
Assignment 1
2022/2023 Semester 2

Due Date: February 10, 2023 at 10:00 p.m.
Lockout Date: February 12, 2023 at 10:00 p.m.

Overview:
An object-oriented application is required for modelling a simple luggage management system. The application consists of four domain classes: Flight, LuggageManifest, LuggageSlip, and Passenger.

UML Diagram of Domain Classes

Figure 1 shows a simplified UML diagram of the domain classes along with a main class called LuggageManagementSystem which manages Flight and Passenger objects. A Flight object has one LuggageManifest object which stores many LuggageSlip objects.

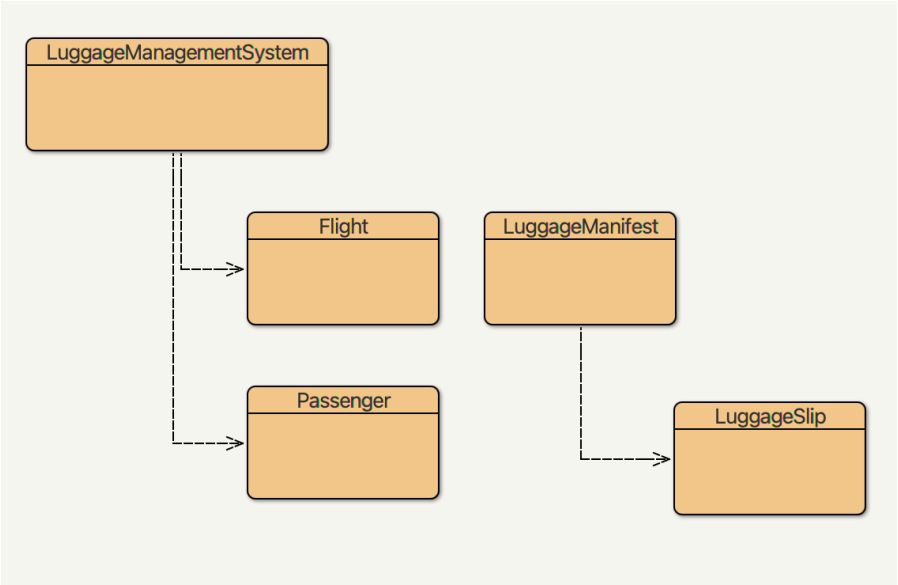

Figure 1: UML Diagram of the Domain Classes

Important Notes: Accessors must be provided for all attributes in all of the domain classes, mutators where appropriate/specified, and information hiding principles must be adhered to (use of access modifiers as appropriate). There are additional methods specified for the classes in the pages that follow. These methods are necessary for auto-testing of your program. Therefore your solution may or may not use these methods directly, however, the methods must be present in your code.

Passenger Class

The **Passenger** class models passenger who checks zero or more pieces of luggage into a flight. It has the following attributes and methods. Accessors are required for all attributes.

Attribute	Type	Purpose
passportNumber	String	The passenger's passport number
flightNo	String	The passenger's flight number
firstName	String	The passenger's first name
lastName	String	The passenger's last name
numLuggage	int	The number of pieces of luggages that the passenger checks in for the flight
cabinClass	Char	Represents the cabin class that the passenger belongs to on the flight. Possible values: 'F', 'B', 'P', 'E'

Method Signature	Return Type	Purpose
Passenger (String passportNumber, String firstName, String lastName, String flightNo)		Constructor. Initialises the state of a Passenger object. The numLuggage and cabinClass variables are set randomly using the appropriate methods in the class.
assignRandomCabinClass( )	void	Sets the cabinClass variable to a random cabin class value : 'F', 'B', 'P', 'E'
toString( )	String	Returns a String representation of a Passenger object. See format below

Format for the toString( ) output for a Passenger, Joe Bean, with passport number TA890789:
PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 3 CLASS: E

LuggageSlip Class

The **LuggageSlip** class models a paper slip that is generated for a piece of luggage when it is checked into a flight for a given passenger. It has the following attributes and methods. Accessors are required for all attributes.

Attribute	Type	Purpose
owner	Passenger	The Passenger who owns/checks in the luggage.
luggageSlipIDCounter	int	A class variable that starts at 1 and increments by 1 for each new LuggageSlip object created.

Attribute	Type	Purpose
luggageSlipID	String	A unique identifier for the LuggageSlip, produced using Passenger's flight number, last name and the luggageSlipIDCounter. Example below
label	String	A string variable for recording any additional details on the luggage slip.

Method Signature	Return Type	Purpose
LuggageSlip (Passenger p, Flight f)		Constructor. Initialises all state using the input parameters, and sets the label to an empty String.
LuggageSlip (Passenger p, Flight f, String label)		Overloaded constructor. Initialises all state using the input parameters (including label).
hasOwner (String passportNumber)	boolean	Return true if the owner of the LuggageSlip has the supplied passportNumber, false otherwise.
toString( )	String	Returns a String representation of the LuggageSlip object. See format below.

Sample luggageSlipID values for a Passenger, Joe Bean, with three pieces of luggage on flight BW600:
BW600_Bean_1
BW600_Bean_2
BW600_Bean_3

Format for the toString( ) output for a LuggageSlip for Joe Bean
BW600_Bean_1 PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 3 CLASS: E $105
BW600_Bean_2 PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 3 CLASS: E $105
BW600_Bean_3 PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 3 CLASS: E $105

The additional token at the end ($105) is the label (if set).

LuggageManifest Class

The **LuggageManifest** class manages one or more LuggageSlip objects within a slips collection. Accessors are required for all attributes.

Attribute	Type	Purpose
slips	ArrayList<LuggageSlip>	Stores multiple LuggageSlip objects.

Method Signature	Return Type	Purpose
LuggageManifest ( )		Constructor. Initialises the slips collection.
addLuggage(Passenger p, Flight f)	String	Adds one or more new LuggageSlip objects to the slips collection based on the number of pieces of luggage that a Passenger has. This method checks the number of allowed pieces based on the Passenger's cabin class and calculates the cost of excess luggage. The excess luggage cost is then added as a label on all luggage slips for the Passenger. The method returns the output indicated below*.
getExcessLuggageCost(int numPieces, int numAllowedPieces)	double	Calculates and returns the total cost of adding excess luggage based on the number of allowed pieces. Every excess piece of luggage costs $35.00.
getExcessLuggageCostByPassenger(String passportNumber)	String	Returns the total cost of excess luggage (if any) on the manifest for a Passenger with a given passport number or "No Cost" otherwise.
toString( )	String	Returns a String representation of the aggregated state of a LuggageManifest by traversing and returning the String representation of each LuggageSlip (if present).

*Sample output for the addLuggage(..) method for Passenger 1 - 2 pieces of luggage, 1 excess
PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 2 CLASS: P
Pieces Added: (2). Excess Cost: $35

*Sample output for the addLuggage(..) method for Passenger 2 - 0 pieces of luggage
PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 0 CLASS: P
No Luggage to add.

*Sample output for the addLuggage(..) method for Passenger 3 - 1 piece of luggage, 0 excess
PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 1 CLASS: B
Pieces Added: (1) Excess Cost: $0

Sample toString( ) output for the LuggageManifest :

LUGGAGE MANIFEST:
BW600_Bean_1 PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 2 CLASS: F
BW600_Bean_2 PP NO. TA890789 NAME: J.BEAN NUMLUGGAGE: 2 CLASS: F
BW600_Deer_3 PP NO. BA321963 NAME: L.DEER NUMLUGGAGE: 2 CLASS: P $35
BW600_Deer_4 PP NO. BA321963 NAME: L.DEER NUMLUGGAGE: 2 CLASS: P $35

Flight Class

The **Flight** class models a flight with the following attributes and methods. Accessors are required for all attributes.

Attribute	Type	Purpose
flightNo	String	Six character code for a Flight
destination	String	The airport code for the Flight's arrival
origin	String	The airport code for the Flight's departure
flightDate	LocalDateTime	The date and time of the Flight
manifest	LuggageManifest	The LuggageManifest for the Flight that stores all of the LuggageSlips created for checked-in luggage

Method Signature	Return Type	Purpose
Flight (String flightNo, String destination, String origin, LocalDateTime flightDate)		Constructor. Initialises the state variables, with the supplied parameters, as applicable, and creates a new LuggageManifest object
checkInLuggage(Passenger p)	String	Validates whether a Passenger can check in luggage for the flight (same flight number as the Passenger's).If this is true, the method adds the Passenger's luggage to the flight using the addLuggage(..) method and returns the String outcome. If the Passenger's flight number does not match the Flight, then the message "Invalid flight" is returned.
printLuggageManifest( )	String	Returns a String representation of the manifest
getAllowedLuggage(char cabinClass)	int	A class method that returns the number of allowed pieces of luggage for a given cabin class, after which a cost is incurred. 'F': 3 pieces 'B': 2 pieces 'P': 1 piece 'E': 0 pieces
toString( )	String	Returns a String representation of the Flight.

Sample toString( ) output for a Flight:
BW600 DESTINATION: POS ORIGIN:YYZ 2023-01-23T10:00

LuggageManagementSystem

This program should read Flight and Passenger data from a file. A sample is provided below that shows a few simple test cases. You are required to test as many cases as possible.

```
import java.time.LocalDateTime;

public class LuggageManagementSystem
{
    public static void main( String[] args ){
      //You are required to create Flights and passengers from file
      LocalDateTime d = LocalDateTime.of(2023,1,23,10, 00, 00);
      Flight yyz = new Flight("BW600", "POS", "YYZ", d);

      System.out.println(yyz);
      Passenger p ;
      String[] pps = {"TA890789", "BA321963", "LA445241"};
      String[] firstNames = {"Joe", "Lou", "Sid"};
      String[] lastNames =  {"Bean", "Deer", "Hart"};

      for(int i = 0; i<3; i++){
          p = new Passenger(pps[i], firstNames[i], lastNames[i], "BW600");
          System.out.println(p);
          System.out.println(yyz.checkInLuggage(p));

      }
      System.out.println(yyz.printLuggageManifest());
    }
}
```

Submission Instructions

- Write the Java code for each class in the application using BlueJ.
- Document your student ID at the top of each file within a comment block.
- Upload a ZIP file of your compiled project source and class files to the myElearning course page by the deadline. Submissions that do not compile will receive 0 marks.
- Name your ZIP file as follows: **FirstName_LastName_ID_A1.zip.** Marks will be deducted for submissions that do not conform to this naming convention.
- Sign and submit the University Plagiarism declaration confirming that you are submitting your own original work and that you have not copied or collaborated with other students.
- Early submission points can be earned if you submit before the due date. No penalties will be applied but no early points will be earned if you submit by the lockout date.

Important Information Regarding Academic Conduct

- This is an individual assignment. You must attempt this assignment by yourself without any help from others.
- You may use legitimate resources on the Internet, in books, or from the course notes to assist (unless prohibited by a question). Copying or modifying such content does not make it yours. Indicate on your submission any or all sources used in your answers within comments at the end of your files. Identify code snippets that you did not write yourself but reused from some source.
- You are not allowed to communicate, share or disclose any aspect of your solutions/work in any form to anyone except for the course lecturer, markers, examiners and tutors.
- You are not allowed to assist others with this assignment.
- University plagiarism and academic misconduct policies apply fully.

- No part of your submission should be made publicly available even after the due date without permission from the course lecturer.
- The use of Chegg.com, CourseHero.com or any tutoring or homework assistance services, online or not, are prohibited.
- If you are faced with extenuating circumstances, contact your lecturer right away. Concessions may be arranged on a case-by-case basis. Be honest with your claims and provide evidence where possible.