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

Due Date: February 11, 2018 at 11:50 p.m.

Overview: An object-oriented application is required to manage information on packages delivered by Paketzustellung Inc, a courier company. The application must provide a simple user interface that allows a user in the company to perform the following operations:

- Add a new package for delivery on a given flight
- Query for a particular package or flight
- List all the packages managed by Paketzustellung Inc's system

The application will consist of three domain classes: **Package**, **Flight**, and **PaketzustellungSystem**. The user interface of the application will be provided by another class called **PaketzustellungConsole**.

UML Diagram of Domain Classes

Figure 1 shows a simplified UML diagram of the **Package**, **Flight**, **PaketzustellungSystem and PaketzustellungConsole** classes. A **Flight** object is related to many **Package** objects. A **PaketzustellungSystem** object manages many **Flight** objects. The **PaketzustellungConsole** class invokes the services of the **PaketzustellungSystem** class.

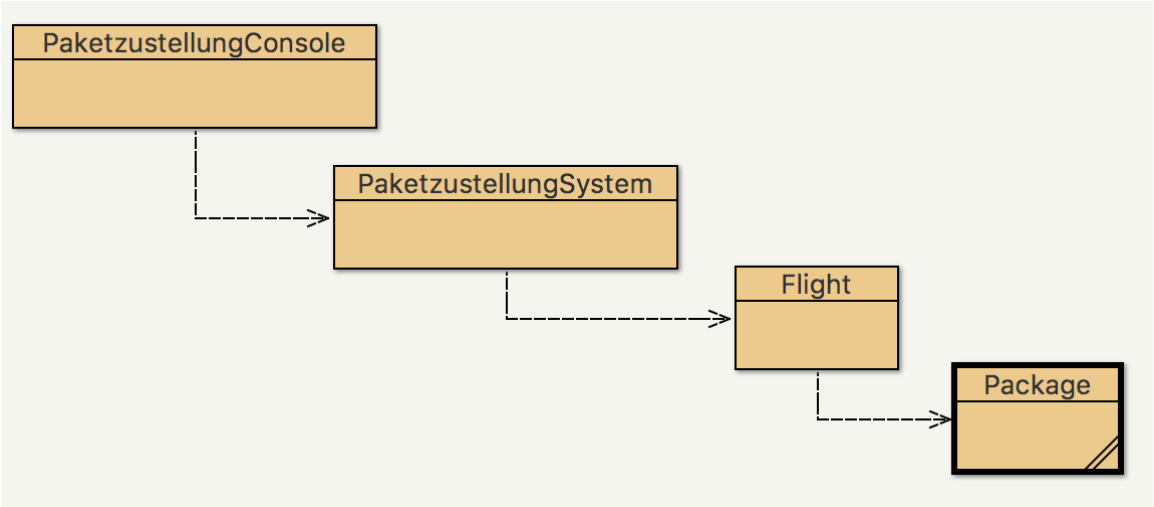

Figure 1: UML Diagram of Domain Classes

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.

Package Class

The **Package** class models a package with the following attributes:

Attribute	Type	Purpose
packageID	int	A unique identifier for a package. This number is automatically generated, starts from 1 and increases by 1 for each subsequent package
weight	double	The weight of the package in kilograms
destination	String	The country where the package will be delivered to
owner	String	The full name of the package's recipient
status	String	The current status of the package. The 3 possible states are: 'In Transit' : if the Package is on a flight 'Delayed': if the Flight is delayed 'Pick up': if the Flight has arrived

The **Package** class has the following methods:

Method Signature	Return Type	Purpose
Package (String destination, String owner, double weight)		Constructor
toString( )	String	Returns a complete description of a Package object's state

Note: Accessors and mutators for the Package attributes should be provided as appropriate

Flight Class

The **Flight** class models a flight with the following attributes:

Attribute	Type	Purpose
flightNumber	String	A unique identifier for a Flight. This identifier takes the format XXYYY where the first two characters are in the range [A..Z] and the last three characters are in the range [0..9]
destination	String	The country where the Flight is going to
packages	Package[ ]	A collection of Packages carried on the flight. A maximum of 10 Packages can be carried.
numPackages	int	The number of Packages currently on the flight
status	String	The current status of the flight. Possible states: On Time (default), Delayed, Arrived.

The **Flight** class has the following methods:

Method Signature	Return Type	Purpose
Flight(String flightNumber, String destination)		Constructor
toString( )	String	Returns a complete description of a Flight object's state
addPackage(String owner, double weight)	boolean	Creates a Package object and adds it to the Flight's package collection.
getPackageDetails(int ID)	String	Searches for a Package with a given ID and returns a String description of the Package, or null otherwise
getPackageDetails(String owner)	String	Searches for Packages owned by a given recipient and returns a String description of all such Packages, or null otherwise
getPackageManifest( )	String	Returns a String description of all Packages carried on the Flight, or null if there are no packages
updatePackageManifest( )	void	Updates the status of the Packages carried on the Flight when the Flight's status changes
updateStatus(char code)	boolean	Updates the status of the flight with the following codes: 'A' : Flight has arrived 'D' : Flight is delayed 'O' : Flight is on time

Note: Accessors and mutators for the Flight attributes should be provided as appropriate

PaketzustellungSystem Class

The **PaketzustellungSystem** class models the following attributes:

Attribute	Type	Purpose
flights	Flight[ ]	A collection of Flights chartered by Paketzustellung Inc to ship packages. A maximum of 5 Flights can be chartered.
numFlights	int	The number of Flights currently chartered.

The **PaketzustellungSystem** class has the following methods:

Method Signature	Return Type	Purpose
PaketzustellungSystem ( )		Constructor
addPackage(String packageDestination, String owner, double weight	boolean	Adds a Package to a Flight in the system that can deliver the Package. Returns true if the Package is successfully added to the system and false in all other cases. Packages are only added to the system if there is a Flight that can accommodate it.

Method Signature	Return Type	Purpose
addFlight(String flightNumber, String flightDestination)	boolean	Creates a new Flight object with a valid flight number and adds it to the collection of Flights chartered. Duplicate Flights are not allowed.
getFlightByDestination(String destination)	Flight	Returns a Flight object with a given destination or null otherwise.
listPackage(int ID)	String	Returns the details of a Package with the given ID in the system or an appropriate message if no package is found.
listPackages(String owner)	String	Returns a list of all Packages in the system across one or more Flights intended for a particular owner or an appropriate message if no packages are found.
listPackages()	String	Returns a list of all Packages in the system or an appropriate message if no packages are found.
listFlights()	String	Returns a list of all Flights in the system or an appropriate message if no flights are found.
updateFlightStatus(String flightNumber, char code)	String	Updates the status of a Flight in the system with a given flight number. Returns a message indicating success or failure

PaketzustellungConsole: User Interface and Main Class

The user interface must enable the user to perform several operations such as:

- Add a new Package to the system
- Add a new Flight to the system
- Display Packages by owner name
- Display a Package's details by ID
- Update a Flight's status (Flight must exist first)
- Display information on all Packages in the system
- Display information on all Flights in the system

The user interface should accept input from the keyboard and generate textual output to the console. The class, **PaketzustellungConsole**, should provide the functionality of the user interface. You should note that the user interface must create an instance of the **PaketzustellungSystem** class before doing anything else. After it receives user input, it forwards requests to the domain classes to accomplish the tasks required. The results are received and displayed on the console.