CSC1016S Assignment 6: UML, Polymorphism, and Inheritance

Assignment Instructions

This assignment involves developing OO models of real world scenarios using UML, and constructing programs in Java using polymorphism, sub classes and inheritance.

Exercise One [40 marks]

This exercise concerns the development of an OO representation for a Passenger Information Point (PIP) in UML.

Scenario

The Takalani Bus Company is developing a real-time information system for passengers. A Passenger Information Point (a PIP) is to be attached to each major stop in the network. Travellers waiting at such a stop will be able to view a list of approaching buses, ordered by estimated arrival time (in minutes).

How does a PIP maintain the arrivals list? At (fairly) regular intervals along each bus route are electronic tags (affixed to street signs and bus stops). When a bus passes one of these waypoints it broadcasts a message to the effect. PIPs are equipped to receive these messages.

A PIP knows about the services that use the stop to which it is attached. Each is characterised by the service identity code, the route number and destination, and a "timetable". The timetable lists the waypoints that a bus must pass as it approaches (those that are "upstream"), and for each, gives the average time it takes a bus to get from there to the stop.

A message consists of three (alphanumeric) codes that uniquely identify the bus, service and waypoint. The bus code is used to determine whether the bus that sent the message is on the arrivals list. If the bus is not on the list, then the service and waypoint codes are used to determine whether it should be added i.e. whether it's a service that uses the PIP's stop and whether it's currently upstream.

The waypoint code is used to look up the estimated arrival time, and the estimated arrival time is used to determine bus position in the arrivals list.

How does a PIP know when a bus has arrived (and can be removed from the list)? It knows the waypoint code for the stop to which it is attached.

As for exercise two, your task is to develop an OO representation. In this case, however, your design must take the form of a UML class diagram; a class diagram that describes a collection of classes that together provide the functionality described in the brief.

In this case, we don't have a precise starting point either. (For exercise two we asked you to start with an Employee class.) To identify possible classes, we recommend that you begin with a noun-verb analysis of the description.

That said, here are a few pointers:

- The boundary between a PIP and the rest of the world is clear: it receives messages from waypoints. It displays an arrivals list.

CONTINUED

- To represent/model the idea that the system receives messages, within your design you should have a class of object (a Receiver perhaps) that has a method that accepts a message as a parameter and that acts upon it.
- You do not need to concern yourself with how the arrivals list is displayed. Just ensure that within the design there is an ArrivalsList class.

You should expect to identify and describe at least 5 classes.

Your solution will be assessed according to (i) how well it supports the required functionality, (ii) its OO qualities, and (iii) use of UML.

Submit your work in your '*Solutions.pdf*' document.

Exercise Two [25 marks]

Write a program to demonstrate the use of inheritance by creating and outputting 3 simple objects, where the classes for the second and third objects inherit from the class for the first object.

The first object is a Vehicle with a number of passengers and colour. The second object is a Car with additional number of doors. The third object is a Plane with a manufacturer and model number. Note: do not inherit Plane from Car!

Use the provided *VehicleDriver.java* and *Vehicle.java* files. Use constructors and override methods as appropriate.

Sample Output:

```
Blue 2 passengers
Black 4 passengers 4 doors
White 416 passengers Boeing 737
```

Exercise Three [35 marks]

On the Vula page for this assignment, you will find a program called PizzaStore.java, that stores details of pizzas (menu item number, size, base, extra cheese, extra garlic), curries (menu item number, size, curry type) and soft drinks (menu item number, size, flavour, bottle or can) in a single array, with the options of listing all menu items or deleting a particular menu item.

To complete the program, further classes are required. Using the details above, and by analysing the PizzaStore.java code and the sample I/O, develop Food, Pizza, Curry and SoftDrink types of object.

Sample Input/Output:

```
Welcome to Great International Food Court
MENU: add (P)izza, add (C)urry, add (S)oft drink, (D)elete, (L)ist,
(Q)uit
p
Enter the menu item number
123
Enter the size
12"
Enter the base
Hand-tossed
Enter extra cheese
Yes
```

CONTINUED

```
Enter extra garlic
Yes
Done
MENU: add (P)izza, add (C)urry, add (S)oft drink, (D)elete, (L)ist,
(Q)uit
c
Enter the menu item number
456
Enter the size
Large
Enter the curry type
Vindaloo
Done
MENU: add (P)izza, add (C)urry, add (S)oft drink, (D)elete, (L)ist,
(Q)uit
s
Enter the menu item number
789
Enter the size
Large
Enter the flavour
Coke
Enter whether it is a bottle or can
Bottle
Done
MENU: add (P)izza, add (C)urry, add (S)oft drink, (D)elete, (L)ist,
(Q)uit
d
Enter the menu item number
456
Done
MENU: add (P)izza, add (C)urry, add (S)oft drink, (D)elete, (L)ist,
(Q)uit
l
Pizza: 123, 12", Hand-tossed, Yes, Yes
Soft Drink: 789, Large, Coke, Bottle
Done
MENU: add (P)izza, add (C)urry, add (S)oft drink, (D)elete, (L)ist,
(Q)uit
q
```

Marking and Submission

Submit the *Solutions.pdf, Car.java, Plane.java, Food.java, Curry.java, Pizza.java* and *SoftDrink.java* files contained within a single .ZIP folder to the automatic marker. The zipped folder should have the following naming convention:

yourstudentnumber.zip

END

CONTINUED