CSC1016S Assignment 4: Classes, Modeling and UML

Assignment Instructions

This assignment involves constructing programs in Java using object composition (i.e. write class declarations that define types of object that contain and manipulate other objects), and involves developing OO models of real world scenarios using UML.

Exercise One [35 marks]

The inspiration for this exercise comes from assignment 2's Cash Register problem. In your solution you may have stored amounts as type *double*. That's not ideal for something that requires 'exactness'. A better solution is to work with integer amounts, typically in the currency's minor unit e.g. cents in the case of Rand. (Have a look at my solution, it's on the Vula assignment page.)

To go a step further, however, integers still aren't a good representation of money because we can do things with integers that don't make sense with money e.g. 45^2 is 2025, but (45 cents)^2? Money squared?

In this exercise we sort the problem out by making Money and Currency classes. (Turns out that 'money is ~~no~~ an object'!) You will find the specifications for the classes on the following pages.

We recommend that you begin with the Currency class. You will find test harness classes on the Vula page. (*TestCurrency.java* and *TestMoney.java*.)

To give a feel for how the components function, here are a couple of code snippets to illustrate.

First, the Currency class:

```
Currency rand =  new Currency("R", "ZAR", 100);
String s = rand.format(30511);
long a = rand.parse(s);
System.out.println(s);
System.out.println(a);
```

The code creates a Currency object to represent the South African Rand. It uses the object to create a String representation of the cents amount 30511, and it uses the object to translate the String back into a cents amount. Finally, it prints the String and the cents amount. Here's the expected output:

```
R305.11
30511
```

Now here is the Money class (assume that we are carrying on from the snippet above):

```
Money mOne = new Money("R14.50", rand);
Money mTwo = new Money("R450.00", rand);
Money mThree = mOne.add(mTwo);
System.out.println(mThree.toString());
```

The code creates two Money objects, one to represent R14.50, and the other to represent R450. It adds them together to get a third money object, and it uses the 'toString()' method to get a String representation of the sum that it then prints out. Here's the output:

```
R464.50
```

NOTE: A Money object will use its associated Currency object to perform String translations.

CONTINUED

Class Money

An object of this class represents an amount of money in a particular currency. Amounts can be added and subtracted.
The amount is stored as a quantity of the minor unit of the currency e.g. 1 Rand will be stored as 100 cents.

Instance variables

private Currency currency;
private long minorUnitAmount;

Constructors

public Money(String amount, Currency currency)

> *// Create a Money object that represents the given amount of the given currency.*
> *// The String is assumed to have the following format: <currency symbol><quantity of*
> *// units>.<quantity of minor units> e.g. in the case of USD, $50.30, $0.34.*

public Money(long minorUnitAmount, Currency currency)

> *// Create a Money object that represents the given amount of minor units of the given currency.*

Methods

public long longAmount()

> *// Obtain a long integer that represents this Money object's value as a quantity of the minor unit of*
> *// the currency. For example, if the Money object represents £1.10 (GBP), then this method will*
> *// produce the long integer 110.*

public Currency currency()

> *// Obtain the currency of this Money.*

public Money add(Money other)

> *// Add the other amount of money to this amount and return the result. The objects must be of the*
> *// same currency.*

public Money subtract(Money other)

> *// Subtract the other amount of money from this amount and return the result. The objects must be*
> *// of the same currency.*

public String toString()

> *// Obtain a string representation of the monetary value represented by this object e.g. "€45.10" for*
> *// a Money object that represents 45 euros and 10 cents.*

CONTINUED

Class Currency

An object of this class represents a Currency such as US Dollars or British Pound Stirling.
A currency has an ISO 4217 currency code and a symbol denoting the currency's major unit. Many currencies have a minor (or fractional) unit such as the cent in the case of US dollars.

A currency object provides facilities for creating strings and for interpreting strings that represent amounts of the currency.

It is assumed that the currency symbol always appears in front of an amount; that negative amounts are represented by a minus sign, '-', that precedes the currency symbol, "-£34.50" for example; that the decimal point is always represented using a full stop; that no attempt is made to group the digits of large quantities, so for example, one million Rand is assumed to be represented as "R1000000" (as opposed to "R1,000,000").

Instance variables

```
private String symbol;
private String code;
private int minorPerMajor
```

Constructors

```
public Currency(String symbol, String code, int minorPerMajor)
        // Create a Currency object that represents the currency with the given unit symbol (e.g. "£" for
        // Sterling), ISO 4217 code, and number of minor units per major units (e.g. 100 in the case of
        // pennies per British Pound).
```

Methods

```
public String symbol()
        // Obtain the symbol or sign for this currency.

public String code()
        // Obtain the ISO 4217 code for this currency.

public int minorPerMajor()
        // Obtain the number of minor units per major unit e.g. there are 100 cents to the Euro.

public String format(long amount)
        // Obtain a string representation for the amount given.
        // The amount is assumed to be in the currency's minor unit e.g. pennies for Stering, cents for Rand.

public long parse(String amount)
        // Obtain a numerical value for the quantity represented by the given string.
        // The result is given as a quantity of the currency's minor unit.
        // The String is assumed to have the format: [-]<symbol><quantity of units>.<quantity of minor
        // units>. (The square brackets indicate that the minus sign is optional.)
```

CONTINUED

Exercise Two [30 marks]

This exercise concerns the development an OO representation of the employees of the Fynbos Flower Company from the point of view of timekeeping. Consider the following English language description:

Scenario

The Fynbos Flower company is flexible on the hours that employees work, and as a consequence, timekeeping is an important administrative practice. The company records the dates and times of shifts (for want of a better word). Employees sign-in when they arrive and sign-out when they leave.

This information supports a range of enquiries

- whether an employee is present,
- whether an employee was present on a given day,
- the details of the shift worked on a given day,
- the total hours worked during a given week.
- the shifts worked during a given week.

The concept of a "week" bears elaboration. A business week starts on a Monday and ends on a Sunday. The weeks are numbered. The first week of the year is the one that contains the first Thursday of the year.

Your task is to develop a specification for an Employee class along with whatever supporting classes are found necessary (such as a Shift class).

- Your specifications should be in the form commonly used for assignment exercises e.g. like those of Money and Currency in exercise one.
  - Give instance variable declarations.
  - Give signatures of constructors and descriptions of behaviour.
  - Give method signatures and descriptions of behaviour.
- Aim for a **RICH** representation – think about exercise one, which aims for a richer representation of Money that that provided by integers or double.

Start with the Employee class and think about the variables and methods required to support the enquiries listed.

- What are the responsibilities of an object of this class?
- How are responsibilities fulfilled? *This question will lead to other classes and other responsibilities.*

NOTE: Each Fynbos employee is associated with a collection of shifts. You will need some type of collection object. Lists and arrays are types of collection, however, you don't need to restrict yourself to these. You may wish to design one that better suits the task at hand.

You should expect to develop specifications for *at least* 4 classes. You should submit your work in the form of a PDF document.

CONTINUED

Exercise Three [35 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.
- 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. Submit your work in the form of a PDF document.

CONTINUED

Marking and Submission

Submit the *Currency.java* and *Money.java* files, the PDF document containing your timekeeping design, and the PDF document containing your Passenger Information Point design contained within a single .ZIP folder to the automatic marker. The zipped folder should have the following naming convention:

yourstudentnumber.zip

END

CONTINUED