CSE1IOO/CSE4IOO Semester 2, 2020
Programming Assignment – Part 1 (20%)

Assessment: This assignment is worth 20% of the final mark for this subject. Please read the entire specification carefully, before you start writing the code.

Due Date: Friday, 4 September 2020, at 10 am.

Delays caused by computer downtime cannot be accepted as a valid reason for a late submission without penalty. Students must plan their work to allow for both scheduled and unscheduled downtime.

This is an individual Assignment. You are not permitted to work as a group when writing this assignment.

Copying, Plagiarism: Plagiarism is the submission of somebody else's work in a manner that gives the impression that the work is your own. The Department of Computer Science and Information Technology treats academic misconduct seriously. When it is detected, penalties are strictly imposed. Refer to the unit guide for further information and strategies you can use to avoid a charge of academic misconduct. ***All submissions will be electronically checked for plagiarism using latcs8 server program. If you are new at La Trobe, please complete academic integrity module on LMS to understand what is regarded as an academic misconduct.***

Objectives: The general aims of this assignment are:

- To analyse a problem in an object-oriented manner, and then design and implement an object-oriented solution that conforms to given specifications
- To practise using inheritance in Java
- To practise file input and output in Java
- To make implementations more robust through mechanisms such as exception handling.

Submission Details and Marking Scheme: Instructions on how to submit electronic copies of your source code files from your latcs8 account and a marking scheme overview are given at the end. If you have not been able to complete a program that compiles and executes containing all functionality, then you should submit a program that compiles and executes with as much functionality as you have completed. (You may comment out code that does not compile.)

NOTE: While you are free to develop the code for this assignment on any operating system, your solution must run on the latcs8 system.

NOTE: You can use Arrays or ArrayList or LinkedList of the Java API. If you use arrays, assume that we never have more than 50 patients in the system.

Problem Description

In this assignment, which consists of two parts, you will develop a prototype of a patient record system.

A medical clinic needs a system to keep information about their patients and medical observations about the patients. The information about a patient and concepts of medical observation is explained below.

- Each patient has a unique patient ID and a name. For each patient, various kinds of medical observations are recorded.
- A medical observation can be quantitative measurements such as height, weight, blood pressure, etc. These measurements are referred to as "measurement observations".
- In addition, there are observations that are of non-quantitative nature, for example, the patient's blood type. These observations are referred to as "category observations".
- Each observation type has a name (e.g. "Height", "Blood type") and a unique code.
- Additionally, each measurement observation type has a unit associated with it. For example, the unit for the 'Blood Pressure' measurement observation type can have the value 'PSI'. Similarly, 'Weight' measurement observation type can have unit value 'kg', etc.
- Each category observation type has a number of valid categories associated with. Each category of a category observation type is specified by a name. For example, the categories for 'Blood Type' category observation type can have values such as, 'Group O', 'Group A', etc.

The nurses and doctors who use the system should be able to:

- Add a measurement observation type
- Add a category observation type
- Add a patient
- Add a measurement observation (for a patient)
- Add a category observation (for a patient)
- Retrieve details of an observation type (given its code)
- Retrieve a patient record by the patient id (including the patient's observations)

There will be additional features, which we will add to our application in part 2 of the assignment.

Description of classes:

Each class described below, should have appropriate constructors, get/set and toString() methods (where applicable) to support different operations. Additionally, some of the necessary attributes and methods are listed below:

The *Patient* class:

- Attributes to store a patient's ID and Name.
- An array to store observations about the patient. The elements in the array are references of type 'Observation'.
- A method to add an observation for the patient. The method must validate the functional correctness (described later) before adding an observation.
- A method to update the value of an observation for a patient.

The *ObservationType* abstract class:

- Attributes to store code and name.

The *MeasurementObservationType* class, which extends the *ObservationType* class:

- Attribute to store Unit.

The *CategoryObservationType* class, which extends the *ObservationType* class:

- Attribute to store Categories.

The *Observation* abstract class:

- Attribute to store the type of observation, which will be of type 'ObservationType'.

The *MeasurementObservation* class, which extends *Observation* class:

- Attribute to store the value of the observation.

The *CategoryObservation* class, which extends *Observation* class:

- Attribute to store the value of the observation.

The *PatientRecordSystem* class:

- An array to store collection of observation types. The elements in the array are references of type 'ObservationType'.
- An array to store collection of patients. The elements in the array are references of type 'Patient'.
- Method to add a measurement observation type.
- Method to add a category observation type.
- Method to add a patient.
- Method to add a measurement observation for a patient.
- Method to add a category observation for a patient.
- Method to retrieve details of an observation type.
- Method to retrieve details of a patient.

To support testing, the *PatientRecordSystem* should also have a toString method to display all the objects stored in the system. The class should be implemented in such a way, so that the test program (*PatientRecordSystemTester*) provided with the assignment, runs without any changes. The Class should NOT take any input from the user via the keyboard.

Array Sizes:

Assume that we can have a maximum of 50 Observation types and a maximum of 50 patients in our system. Also, the number of observations that you can record for a patient cannot exceed 50.

Functional Correctness:

Your classes are required to ensure that the following conditions are met:

1. No two observation types can have the same code
2. No two patients can have the same IDs
3. A patient can have at most one observation of a particular type
4. Observations and their associated observation types must be compatible. For example, a category observation of a patient must be associated with a category observation type and the observation value must be one of the categories of that associated observation type.

Whenever these conditions are violated, an exception of type Exception must be thrown with an appropriate error message.

Your tasks:

To help you along the way, the initial skeletons of the classes have been provided with your assignment. Start working with these classes and add appropriate codes to complete the assignment.

Task 1:

Implement the classes: `Patient, ObservationType, MeasurementObservationType, CategoryObservationType, Observation, MeasurementObservation, and CategoryObservation.`

Task 2:

Implement the `PatientRecordSystem` class, that manages the data of the patients and their records and support the specified operations. Test your code using the test program (`PatientRecordSystemTester.java`) provided with the assignment files.

Task 3:

Make your classes robust so that when a functional correctness is violated, an exception is thrown with an appropriate error message.

Electronic Submission of Your Source Code

- Submit all your Java files on latcs8 server.
- The code has to run under Unix on the latcs8 machine.
- If you want to copy a file from your local machine (laptop, desktop at home) to latcs8 server, use WinSCP (for Windows), or Cyberduck (for mac).
- While submitting the files on latcs8, make sure you are in the same directory as the files you are submitting.
- Submit each file separately using the submit command. For example, for a file called 'Patient.java', use the following command:

  ```
  submit IOO Patient.java
  ```

- After submitting the files, you can run the following command to check which files you have submitted:

  ```
  verify
  ```

- You can submit the same file as many times as you like before the assignment deadline. The previously submitted copy will be replaced by the latest one.

Marking Scheme Overview

The assignment has the total of 100 marks, which are distributed as follows:

- Implementation (Execution of code) 90 marks (Do all parts of the programs execute correctly? Note your programs must compile and run to carry out this implementation marking. So, comment out the non-working code in your submission.)
- Code Design, Layout and Documentation 10 marks (Does the program conform to specifications? Does the program solve the problem in a well-designed manner? Does the program follow good programming practices? Does the indentation and code layout follow a good, consistent standard? Are the identifiers meaningful? Are comments useful in explaining what and how the program works? (Javadoc comments are optional.)
- A detail marking rubric will be published on LMS soon.

END