ASSIGNMENT PART 1

Individual Assignment: 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 FPT university treats academic misconduct seriously. When it is detected, penalties are strictly imposed.

Objectives: The general aims of this assignment are:

- To analyze 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.

Problem Description

In this assignment, which consists of two parts, you will develop a prototype of a patient record system. The description below applies to both part 1 and part 2. The tasks required for part 1 will be described later in this handout.

A medical clinic needs a system to keep information about their patients and medical observations about the patients. The concept 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 measurement 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.
- Each measurement observation type has a unit associated with it. Each unit is identified by a name.
- Each category observation type has a number of valid categories associated with.

Each category of a category observation type is specified by a name. The nurses and doctors who use the system should be able to:

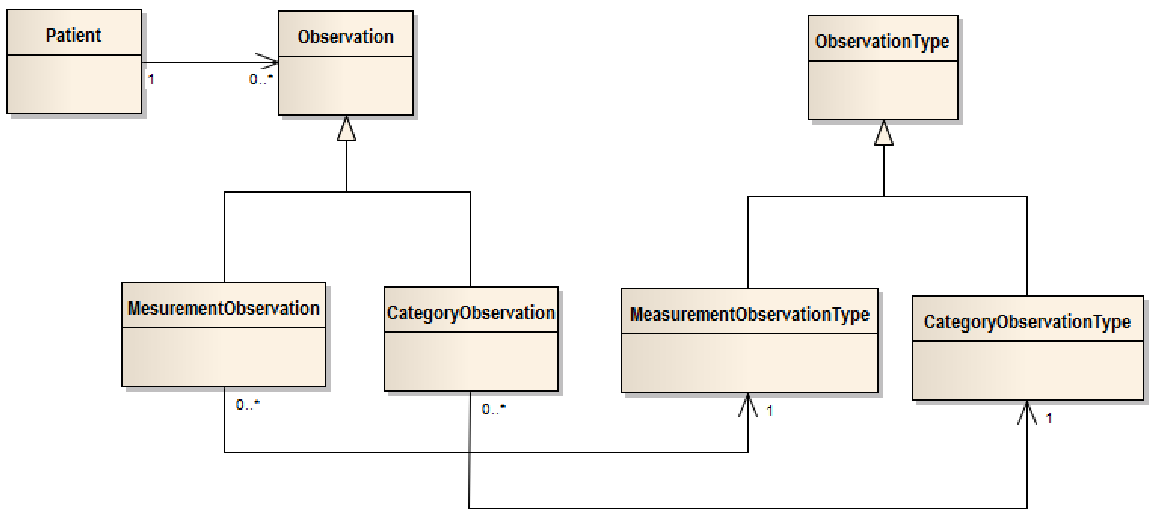
- Add a measurement observation type
- Add a category observation type
- Delete an observation type (which has no associated observations)
- Add a patient
- Add a measurement observation (for a patient)
- Add a category observation (for a patient)
- Modify the value for a measurement observation
- Modify the category for a category observation
- Delete an observation (of a patient)
- Delete a patient (delete all the patient's observation as well)
- Retrieve details of an observation type (given its code)
- Retrieve a patient record by the patient id (including the patient's observations)

In addition (though we will not be concerned with these for Part 1), the nurses and doctors should be able to

- Save all the data
- Load data from the file

Design Class Diagram

A design has been made of the system and is presented in the class diagram below.

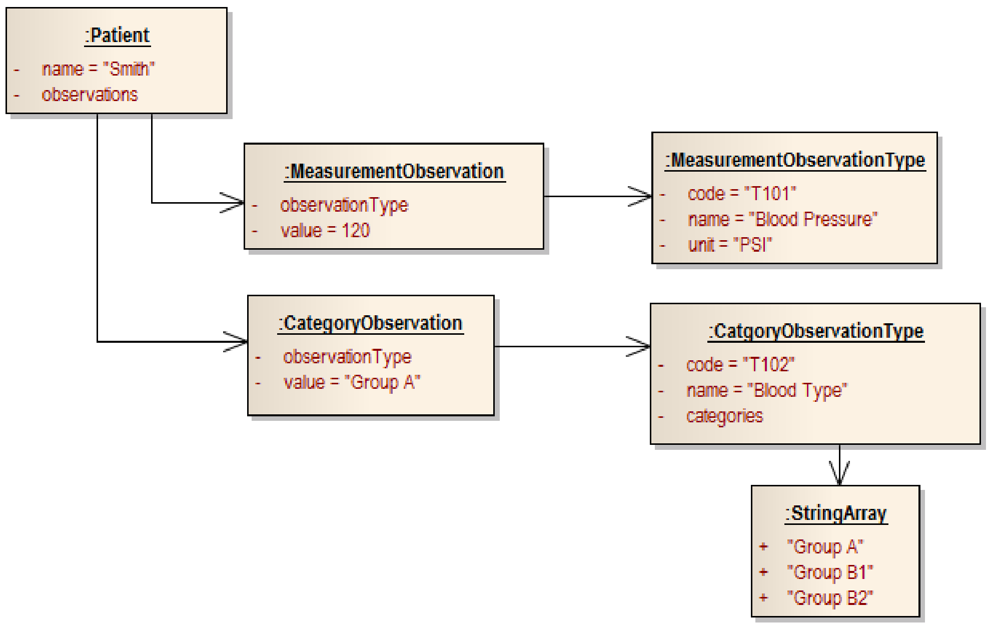


Note: The different shapes of the arrows denote different relationships. The empty triangular arrow denotes the **'is-a'** relationship, as shown in lectures. The other type of arrow denotes the **association** relationship. For example, the arrow from Patient to Observation means that the two classes are associated with each other. Moreover, with the arrow going from Patient to

Observation, it means that a Patient object has references to Observation objects associated with the Patient.

A snapshot of the objects

Note that each observation must be associated with an observation type. Thus, a snapshot of a patient's data may look like what shown in the diagram below.



Task 1 – Implementing all the classes required to validate the design class diagram

As the first step in the implementation process, it is desirable to validate the design class diagram.

Toward this purpose, for Task 1, you are required to

- Implement the classes in the design class diagram: **Patient**, **ObservationType**, **MeasurementObservationType**, **CategoryObservationType**, **Observation**, **MeasurementObservation** and **CategoryObservation**.
  Include all the necessary attributes and methods, which you need to identify.
- Implement a class, called **PatientRecordSystem**, that manages the data of the patients and their records.

Operations required to be supported

For validating the design, the **PatientRecordSystem** class should provide the methods to perform the following operations:

1. Add a measurement observation type

2. Add a category observation type
3. Add a patient (enter details such as id, name)
4. Add a measurement observation (for a patient)
5. Add a category observation (for a patient)

For testing purpose, as will be required for Task 2, the **PatientRecordSystem** should also have a *toString* method to display all the objects stored in the system.

No Interactive Inputs

The **PatientRecordSystem** class must be implemented in such a way that it can be tested by the program given in the Appendix without any changes. Thus, it should not take any interactive input. That is, it should not take any input by the user via the keyboard.

Array Sizes

Assume that we can have the maximum of 50 observation types, and the maximum of 100 patients.

Functional Correctness

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

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

When those conditions are violated, an exception of type Exception should be thrown.

Task 2 – Testing the PatientRecordSystem class

Test your **PatientRecordSystem** class with the test program provided in the appendix. Your classes should be implemented in such a way that the provided test program can be run without any changes.

Try to make the output of the tests easy to read.