Practical 7: Abstract Classes and Interfaces

Q1(a) The patients of a hospital are categorized as either *inpatient* or *outpatient*. An inpatient is someone who is hospitalized for at least one night whereas an outpatient does not require a hospital stay. The hospital offers 2 types of rooms:

- *'S'* – a *Standard* room which has 4 beds and the room rate is RM100.00 per night, and
- *P'* – a *Private* room which has 1 bed and the room rate is RM200.00 per night.

You are required to create classes to be used in a *patient billing* application. The data to be stored and the calculation of charges for each type of patient are shown below:

Patient type	Data	Calculation of total charges
Outpatient	visit id, name, registration fees, consultation fees	total charges = registration fees (RM30.00) + consultation fees
Inpatient	visit id, name, specialist charges, lab charges, room type, duration of stay	total charges = specialist charges + lab charges + room charges. *where* room charges = duration of stay * room rate

Write the following classes:

- An *abstract class* named `PatientBill`
- The class `OutpatientBill` which is derived from the `PatientBill` class
- The class `InpatientBill` which is derived from the `PatientBill` class

Requirements:

- Define and use *static variables* to represent the registration fees and room rates in the appropriate classes.
- Provide a method to calculate the total charges for the patient bill.
- For the abstract class, define the data fields as `protected` and declare *abstract method*(s) where appropriate.

(b) Write a test program that creates an array of 4 `PatientBill` objects. For each object, display the bill information and also the total charges.

(c) Add a method to your test program named `computeTotalCollection` that takes an array of `PatientBill` objects as parameter and then computes and returns the sum of all the bills.

Q2. Modify your `PatientBill` class from Q1 such that it implements the `Comparable` interface as follows:

Implement the `compareTo` method such that it compares the current object with the parameter object based on the *patients' name*.

Test the `compareTo` method using the `selectionSort` method (to be provided by your lecturer) and observe the differences in the outputs.

Q3. Create an *interface* called `Colorable` with a `void` method called `howToColor`. Create a class called `ComparableCircle` that extends `Circle` class (to be provided by your lecturer) and implements 2 interfaces - `Comparable` and `Colorable`. The `compareTo` method should be implemented by comparing 2 circles on the basis of radius.

Draw the UML diagram and write a test program to find the larger of 2 `ComparableCircle` objects. Display also a message indicating how a `ComparableCircle` object to be colored.