# COMSATS University Islamabad, Lahore Campus

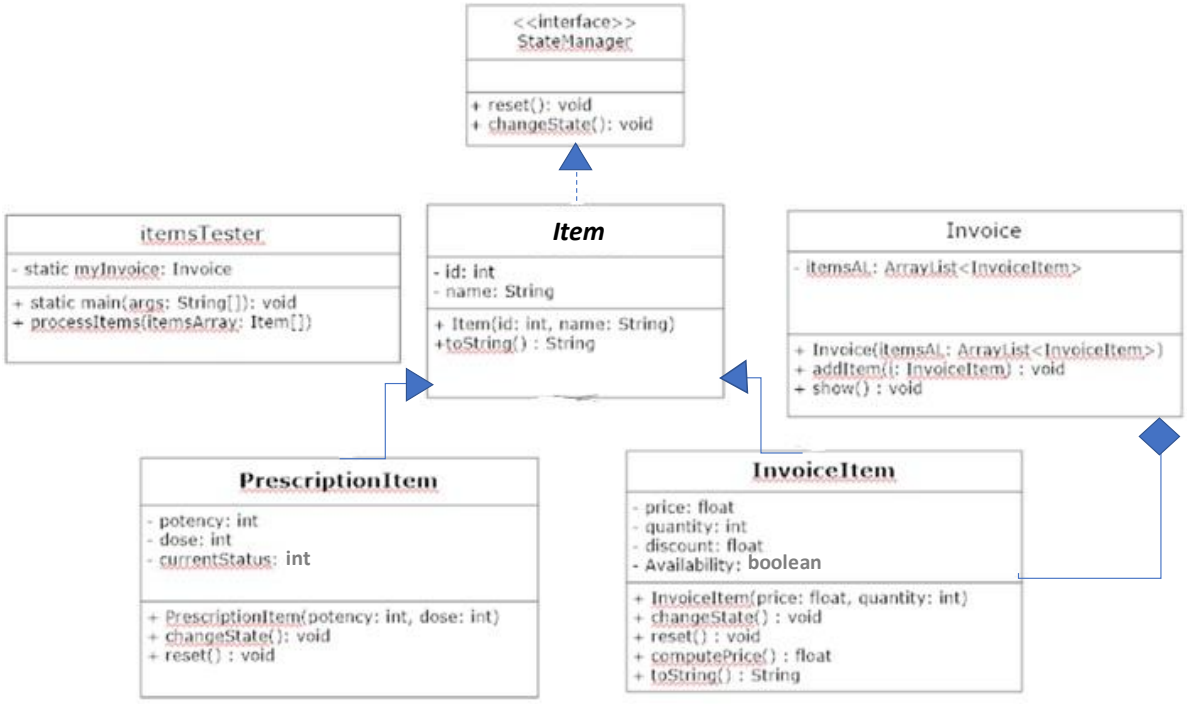
## Final Lab Examination – SPRING 2021

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Course Title: | Object Oriented Programing | | | | Course Code: | CSC241 | Credit Hours: 4(3,1) |
| Course | Dr. Shahbaz Akhtar Abid | | | | Programme Name: | BCS | |
| Semester: | 3rd | Batch: | SP20-BCS | Section: | A | Date: | 22-06-2021 |
| **Time Allowed:** | **100 Minutes** | | | | **Maximum Marks:** | | **50** |
| Student's Name: | | | | | Reg. No. | CUI - - - /LHR | |

**Important Instructions / Guidelines:**

- Consider java language for all answers and return question papers at the end of exam.

**Problem Statement:** Consider the scenario of a Hospital Management System. The following UML Class diagram shows a Hierarchy of some of the classes in the system.

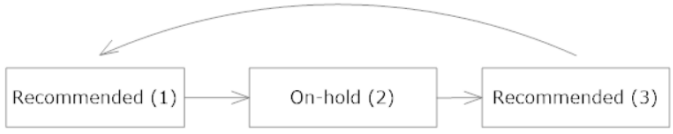



**Concrete Classes and Interface [25 marks]**

- **The StateManager Interface** is an interface, that implements methods for an object having some sort of state. In our example, the InvoiceItem can have state (Availability: "Available" - true, "Unavailable" - false), whereas PrescriptionItem has state (currentStatus : "Recommended" – 1, "On-hold" – 2 or "Prohibited" – 3).
- **The Item class** is self-explanatory. The toString method of Item class prints information in the following format.

  "Item Name: <<name>> (<<id>>)" e.g. Item Name: Panadol (223)

- The constructors of **PrescriptionItem** and **InvoiceItem** initialize fields to appropriate parameters (no validation required), if applicable. Rest of the uninitialized fields are assigned default values, e.g. availability is set to true, and currentStatus is set to 1 (Recommended).
- The **changeState** method in both the classes is implemented differently.
    A. Inside **InvoiceItem**, the state (availability) is toggled between true and false.
    B. Inside **PrescriptionItem**, the state (currentStatus) is switched in the following order:




- **The reset method in both classes** simply initializes the state to default value (as in constructor of each class**).**
- **The computePrice method** returns (does not print) the price of invoice item according to price, quantity and discount rate (percentage).
- **The toString method of InvoiceItem class** prints the invoice item in the following format.

  Item Name: <<name>> (<<id>>)

  Price: <<price>> Quantity: <<quantity>> Total: <<computePrice()>> (@<<discount>> % discount)

  e.g.

  Item Name: Panadol (223)

  Price: 2 Quantity: 150 Total: 270 (@10 % discount)

- **The Invoice class** stores a number of invoice items in an invoice.
    A. The **constructor** simply initializes itemsAL to an empty ArrayList.
    B. **The addItem method** adds one item passed as parameter to itemsAL.
    C. **The show method** prints all invoice items by calling the toString method of each invoice item.

**Driver Class [25]**

- Finally, **The InvoiceTester class** is the test class. It contains an object of Invoice class as **static** field (create the myInvoice object through inline initialization)

  The two methods inside the InvoiceTester class work as described below**.**

    A. **Inside the main method**, create an array 'itemsArray' of class Item containing 3 items. The array should store 2 objects of InvoiceItem and one object of PrescriptionItem. Call the processItems method with itemsArray as parameter.
    B. **The processItems method** performs the following tasks for EACH item in the itemsArray parameter:
        1. If the current item is InvoiceItem, then downcast the item to InvoiceItem and set its discount value to 10 and add to myInvoice by

calling addItem method through myInvoice object. Then, print current Invoice.

2. If the current item is PrescriptionItem, then downcast the item to PrescriptionItem and, write it object to file “PrescriptionItems.ser”.
3. Later, read file Items.ser and display the details of of item with ID = 223.
4. After all elements are processed (the loop is ended), print the invoice data using the show method of myInvoice.

---