Lab01

[IT618, Enterprise Computing, Autumn'23]

Instructor: PM Jat (pm_jat@daiict.ac.in)

The objective of this is to write simple business classes using good OOP principles.

Exercise #1

Create a class `Item`. An item object encapsulates the following fields:

`item_code (integer)`,
`description (string)`,
`qty_in_stock (integer)`,
`min_qty_in_stock (integer)`: it is the minimum quantity required in stock,
`cost (double)`.

Interface of `Item` class.

```
//Constructors
Item(int code,String description,int qty,int min_qty,double cost)
Item(int code, String description, double cost)
//sets qty and min_qty to zero

//Methods
int getItemCode()
String getItemDescription()
int getCost()
void setCost(double cost)
int getStock() //gets Quantity
int getMinQty() //gets min quantity
void addStock(int qty)
//increases the stock by given qty
void withdrawStock(int qty) throws InSufficientStock
//decreases the stock by given qty
boolean isUnderStock()
//returns true if item is under stock, i.e. qty < min_qty otherwise false
```

Also create `ItemTester` that constructs a few item objects, and performs various manipulation operations specified in the above interface.

Deliverables of this exercise: Source Code of `Item` and `ItemTester` classes.

Exercise #2

Create a class `Inventory`.

`Inventory` is basically a collection of `Item` objects. Interface of the `Inventory` class is given below. Functional description of its operations is also given inline with the interface.

Let you use Java `HashMap` for having an in-memory database of items.

Also, create a console-based client program simulating various responsibilities of the `Item` and `Inventory` classes.

Interface of `Inventory` class:

```
//Methods
Item getItem(int itemno) throws ItemNotFound
// returns new object of inventory item after reading data for given item number
from database. It throws the exception if item not found.
void addItem(Item item) throws ItemAlreadyExists
void updateItem(Item item) throws ItemNotFound
void addStock(int item_code, int qty) throws ItemNotFound
void withdrawStock(int item_code, int qty)
        throws ItemNotFound, InSufficientStock
void deleteItem(int item_code) throws ItemNotFound
ArrayList<Item> getAllItems()
//returns all items in inventory.
ArrayList<Item> getItemsUnderStock()
//returns all items that are under stock, i.e. below required minimum stock.
double totalInventoryCost()
//returns total cost of inventory, i.e. summation of cost of all items in the
inventory
```

Deliverables of this exercise: Source Code of `Inventory` and `InventoryTester` classes.