WESTMINSTER
INTERNATIONAL UNIVERSITY IN TASHKENT
An Accredited Institution of the University of Westminster (UK)

Module name and code	Object-Oriented Programming, 5COSC018C
CW weighting	50%
Lecturer setting the task with contact details and office hours	Avaz Khalikov akhalikov@wiut.uz
Submission deadline	June 5th 2024 11:59 PM
Results date and type of feedback	Written, in 2 weeks after deadline
The CW checks the following learning outcomes:	
3. Recommend an extendable and maintainable system architecture based on Object Oriented Programming principles. 4. Identify common cases for Design Patterns in software design. Show ability to apply Design Patterns and justify their use. 5. Analysis of requirements and implementation of clean OOP code.	

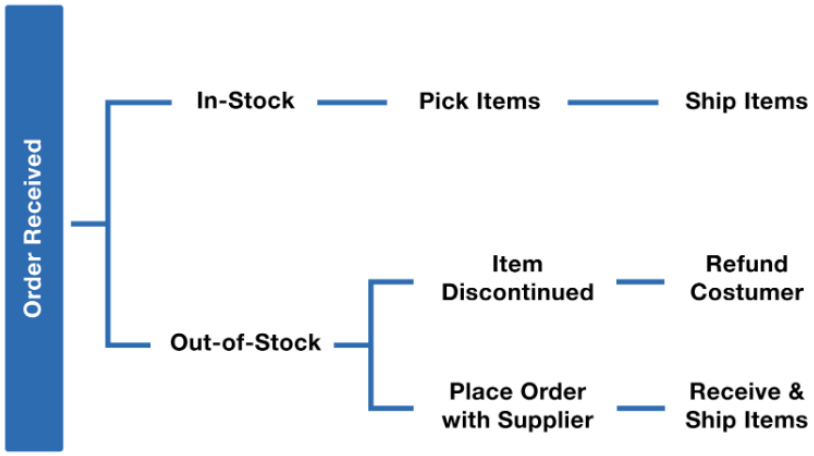
Instructions of Upload: ZIP/Rar archive file must be uploaded into intranet.

Overview

This is an individual piece of work. To accomplish this CW, you are requested to complete all tasks below.

Your task is to implement a simple C# **Product Inventory System** that can be used to add, remove, and update products (CRUD operations), stock management functions and simple reporting, which prints results into the Visual Studio Console.

The task is process based and you must follow the steps and given domain entities architecture.



Task #1
Total: 25 points

1. Create following domain classes with given attributes, follow the naming. Make sure to set the access modifiers accordingly.

name: InvProduct
attributes:
ProductId: A unique identifier for each product.
Name: The name of the product.
Description: A brief description of the product.
Price: The retail price of the product.
StockQuantity: The amount of the product currently in stock.

name: InvCategory
attributes:
CategoryId: A unique identifier for each category.
Name: The name of the category.
Description: A description of what types of products are included in the category.
Suggested methods*: CRUD*

name: SupplierHolder
attributes:
SupplierId: A unique identifier for each supplier.
Name: The name of the supplier.
ContactInformation: Information on how to contact the supplier.
Suggested methods*: CRUD*

name: InventoryRecord:
attributes:
RecordId: A unique identifier for each log entry.
SupplierId: FK from Supplier Holder
ProductId: The product affected.
QuantityChange: The amount by which the product's stock has changed.
Date: The date of the change.
Type: The type of transaction
Status: In Stock, Out of Stock
Suggested methods*: OutofStock(Ids)* – Checks For InvProduct if still remains in Stock
PlaceOrderWithSupplier(Ids) – Makes a request to Supplier to get more InvProducts
OrderReceived(Ids) Adds received InvProduct into Inventory List.
ShipItems(ids) – items get deducted from Inventory List

Total 2 points

2. Create a method called `AddInvProduct()` that takes a `InvProduct` object as a parameter and adds it to the `InvProducts` list, in a similar manner create `AddInvCategory()` method, which adds InvCategory item into `InvCategories` list and a method that returns list of `InvCategories,` and update your Console app Menu.

Total 4 points

Implement Inventory Record methods accordingly: *OutofStock(Id), PlaceOrderWithSupplier(Ids), OrderReceived(Id) and ShipItems(ids)*

Total 4 points

3. Create methods that handle add/remove a product into a InvCategory and returns a list of all products by InvCategory Name and ID. ***(2 point)***

Create a method called `RemoveInvProduct()` that takes a `InvProduct` object as a parameter and removes it from the `InvProduct` list. Create a method called `GetTotalCost()` that returns the total cost of all the InvProducts in the `InvProduct` list. ***(2 points)***

Implement the CRUD operations for **SupplierHolder** class. **(2 points)**

Total 4 points

4. Create UML Diagram(s) of all your classes within the Visual Studio.

Total 5 points

Task #2

Total: 35 points

The above code must implement a simple Product Inventory System that uses the following best practices:

- The `Domain` classes are a well-defined with clear responsibilities(methods). ***(4 points)***
- The methods are well-defined with clear responsibilities. ***(4 points)***
- The code is well-formatted and easy to read. ***(4 points)***
- The code is modular and easy to test. ***(4 points)***
- The code uses appropriate data structures and algorithms. ***(4 points)***
- The code is efficient and uses the least number of resources possible. ***(4 points)***
- Usage of interface ***(4 points)***

<u>Your task:</u>

1. Improve/clean up code in Task #1

2. **Explain for each <u>bullet-point</u> above**, why your code corresponds each best practice.

Update your Console app Menu (InvCategories, InvProducts in InvCategories) to test. (see for reference *Display Hints below) **(7 points)**

Task #3:

Total possible points to get: 20 points.

The goal of the next task is to create an XML Repository based on Inventory Service classes.

1. Implement the `XmlRepository` class. The class should be used to store the Inventory Products in an XML file.

This class implements the following methods:

- `SaveInventoryProducts()`: This method saves the products to an XML file. ***(5 points)***
- `LoadInventoryProducts()`: This method loads the products from an XML file. ***(5 points)***

The `SaveInventoryProducts()` method uses the `XmlWriter` class to write the products to the file. The `LoadInventoryProducts()` method uses the `XmlReader` class to read the products from the file.

This implementation of the `XmlRepository` class is simple and easy to use. It can be used to store and load products in an XML file.

2. Make sure to Save by implementing _repository.SaveInventoryProducts(_products); every time your add or remove a product. ***(5 points)***
3. Load saved products from repository, use in your products list and add necessary comments into the code ***(5 points)***

Task #4:

Total possible points to get: 20 points.

Create a separate project with Repository pattern applied to above project. Name it "StockInventoryRepositoryPattern"

`IRepository` interface defines the methods that are needed to access the products. The InventoryProductRepository class implements the `IRepository` interface and provides a concrete implementation of the methods using an XML file. The StockInventory class uses the `IRepository` interface to access the products. `GetInventoryProducts()` ***(2 points)***, `AddInventoryProduct(InventoryProduct InvProduct)` ***(2 points)***, `RemoveInvProduct (by ID)` ***(6 points)***

Update your Console app Menu (Add Product, Remove and List InvProducts in StockInventory) to test. (see for reference *Display Hints below) ***(5 points)***

Update your Console App Main() method to be able to add, remove, list all product into a Inventory a clean approach, get product by name Menu Items. ***(5 points)***,

You are not required to produce the same Output; the purpose of this Coursework is not just to achieve the following sample output but being able to write the clean code by following OOP principles.

=== *SAMPLE Display Hints ====

***SAMPLE** USER MENU interaction by using Console, please modify it accordingly:*

1. To List Inventory Categories, Enter Keyword: Inventory **Categories**
2. To List All Inventory Products, Enter Keyword: Inventory **Products**
3. **Inventory Management Functions ….**

> Inventory Categories

Inventory Categories section was selected:

1. Clothing
2. Shoes
3. Bags
4. Jewelry
5. Watches

….

Enter a category number to list products that it contains:

> *1*

Clothing Inventory Category was selected:

1. Crewneck T-shirt – Price $50
2. V-neck T-shirt – Price $55
3. Polo shirt – Price $30
4. Graphic tee – Price $45

….

Enter a product number to add into Inventory:

> 3

Polo shirt has been added into your Inventory.

Your Inventory List:

1. Graphic tee – Price $45

2. Polo Shirt – Price $30

Total Price: $75

Enter keyword, “Menu” to go to Main Menu

Enter keyword Remove, to remove a product from Inventory

> Remove

Enter keyword id of a product.

>