### **Problem Statement: Inventory Management System with Notification System**

Design and implement an **Inventory Management System** for a retail organization using the following concepts:

* **Composition** for managing products in the organization.
* **Abstract classes** and **inheritance** for creating different types of suppliers who will subscribe to the organization for inventory notifications.
* **Notification system** that alerts suppliers when a product's stock level falls below the threshold.

### **Requirements:**

#### **1. Organization Class (Composition of Products):**

* Create an **Organization** class that will hold a collection of Product objects. This can be done using **composition**, meaning that the Organization class will **contain** a vector (or list) of Product objects.
* The Organization class should be able to:
  + Add new products to the inventory.
  + Remove products from the inventory.
  + Search for products by productID or productName.
  + Display the full inventory, including the product name, category, and stock levels.

#### **2. Product Class:**

* The Product class should contain the following attributes:
  + productID (unique identifier)
  + productName
  + category
  + price
  + stockLevel (current stock of the product)
  + reorderThreshold (threshold below which restocking is needed)
* It should have methods to:
  + Update stockLevel when products are sold.
  + Check if restocking is necessary (i.e., if stockLevel is below reorderThreshold).

#### **3. Abstract Supplier Class:**

* Define an **abstract class Supplier** with the following:
  + An abstract method notifyRestock(productID, quantity) that will be called when a product’s stock falls below its threshold.
  + A method subscribeToOrganization(organization) that will allow suppliers to subscribe to an Organization to receive notifications when products need restocking.

#### **4. Concrete Supplier Classes:**

* Implement at least **two concrete classes** that inherit from the Supplier abstract class. These could be:
  + LocalSupplier - A supplier who operates locally and provides immediate restocking.
  + GlobalSupplier - A supplier who might take longer to restock the product.
* Both supplier classes should implement the notifyRestock() method to print a message to the console or simulate an action when a notification is received.

#### **5. Restocking Logic in Organization:**

* In the Organization class, when a product’s stock level drops below its reorderThreshold, the organization should:
  + Automatically notify all subscribed suppliers (i.e., all objects of Supplier subclasses).
  + The notification should include the productID and the quantity needed to bring the product back to its original stock level.

#### **6. Inventory Report:**

* Create a method in the Organization class to generate an inventory report. This report should include:
  + A list of all products, their productName, category, price, and stockLevel.
  + A list of products whose stockLevel is below the reorderThreshold and the quantity that needs to be restocked.

**Example Usage:**

**# Create some products**

product1 = Product(101, "Laptop", "Electronics", 1000, 10, 5)

product2 = Product(102, "Smartphone", "Electronics", 500, 3, 5)

**# Create the Organization and add products**

organization = Organization()

organization.add\_product(product1)

organization.add\_product(product2)

**# Create suppliers**

local\_supplier = LocalSupplier("Local Supplier")

global\_supplier = GlobalSupplier("Global Supplier")

**# Suppliers subscribe to the organization**

local\_supplier.subscribe\_to\_organization(organization)

global\_supplier.subscribe\_to\_organization(organization)

**# Simulate a sale and update stock level**

product1.stockLevel = 4 # below threshold, should trigger restocking

**# Generate inventory report**

organization.generate\_report()

### **Submission Instructions:**

* **Deadline:** 3 days from today.
* Submit a GitHub repository or a zip file containing the following:
  + Complete code with well-defined classes (Organization, Product, Supplier, etc.).
  + A README file explaining:
    - The design choices made.
    - Instructions for running and testing the system.
    - Any additional features or challenges faced while implementing the system.