# Exercise 1: Inventory Management System

Scenario:  
Developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

## 1. Understand the Problem:

- Data structures and algorithms are essential for handling large inventories because they ensure efficient searching, updating, and storage of items.  
- Suitable data structures for this problem include:  
 • ArrayList - for ordered access and traversal.  
 • HashMap - for fast access, insertion, and deletion based on productId.

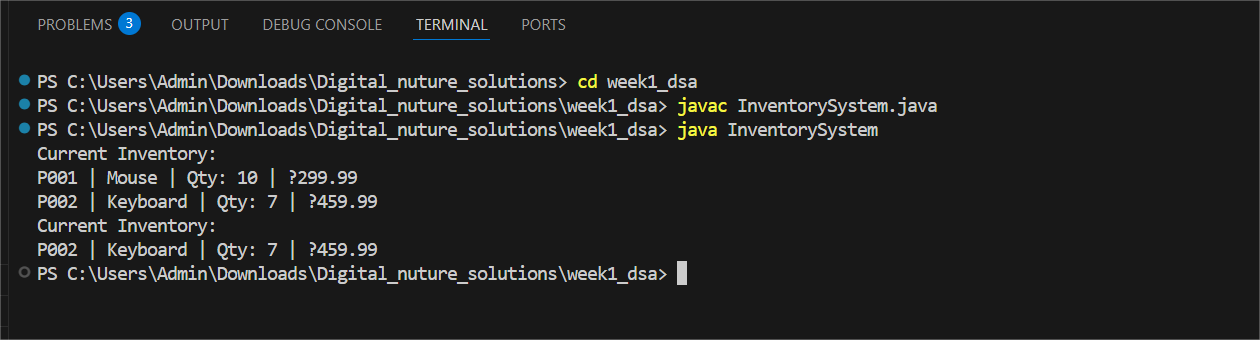
## 2. Setup:

- Create a new Java project for the inventory management system.

## 3. Implementation:

Java Code:

import java.util.\*;  
  
class Product {  
 String productId;  
 String productName;  
 int quantity;  
 double price;  
  
 public Product(String productId, String productName, int quantity, double price) {  
 this.productId = productId;  
 this.productName = productName;  
 this.quantity = quantity;  
 this.price = price;  
 }  
  
 public String toString() {  
 return productId + " | " + productName + " | " + quantity + " | $" + price;  
 }  
}  
  
class InventorySystem {  
 HashMap<String, Product> inventory = new HashMap<>();  
  
 public void addProduct(Product product) {  
 inventory.put(product.productId, product);  
 }  
  
 public void updateProduct(String productId, int quantity, double price) {  
 if (inventory.containsKey(productId)) {  
 Product product = inventory.get(productId);  
 product.quantity = quantity;  
 product.price = price;  
 }  
 }  
  
 public void deleteProduct(String productId) {  
 inventory.remove(productId);  
 }  
  
 public void displayInventory() {  
 for (Product p : inventory.values()) {  
 System.out.println(p);  
 }  
 }  
  
 public static void main(String[] args) {  
 InventorySystem system = new InventorySystem();  
 system.addProduct(new Product("P001", "Mouse", 10, 299.99));  
 system.addProduct(new Product("P002", "Keyboard", 5, 499.49));  
 system.updateProduct("P002", 7, 459.99);  
 system.displayInventory();  
 system.deleteProduct("P001");  
 system.displayInventory();  
 }  
}  
OUTPUT:



## 4. Analysis:

- addProduct: O(1) using HashMap.  
- updateProduct: O(1) since accessing by key in HashMap is constant time.  
- deleteProduct: O(1) as deletion by key in HashMap is also constant time.  
- Optimization: HashMap already provides optimal performance. For sorted inventories, consider TreeMap (O(log n)).