**Exercise 1: Inventory Management System**

**Scenario:**

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

**Steps:**

1. **Understand the Problem:**
   * Explain why data structures and algorithms are essential in handling large inventories.
   * Discuss the types of data structures suitable for this problem.
2. **Setup:**
   * Create a new project for the inventory management system.
3. **Implementation:**
   * Define a class Product with attributes like **productId**, **productName**, **quantity**, and **price**.
   * Choose an appropriate data structure to store the products (e.g., ArrayList, HashMap).
   * Implement methods to add, update, and delete products from the inventory.
4. **Analysis:**
   * Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.
   * Discuss how you can optimize these operations.

1.**Understand the Problem**

**Why are Data Structures and Algorithms Important?**

Handling large inventories involves:

* **Fast access** to product details
* **Quick updates** for quantity/price
* **Efficient addition/deletion** of products

Using optimal **data structures and algorithms** ensures:

* Reduced **search/update time**
* Better **scalability**
* Lower **memory usage**

**Suitable Data Structures:**

* **ArrayList**: Simple, ordered list of products. Slower for searching by ID.
* **HashMap<String, Product>**: Best for fast lookups using productId as the key.  
  ✅ Best choice for inventory where productId is unique.

**CODE:-**

import java.util.\*;

// Product Class

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 void display() {

        System.out.println("ID: " + productId + ", Name: " + productName +

                ", Quantity: " + quantity + ", Price: $" + price);

    }

}

// Inventory Manager using HashMap

class InventoryManager {

    private Map<String, Product> inventory = new HashMap<>();

    // Add Product

    public void addProduct(Product product) {

        if (inventory.containsKey(product.productId)) {

            System.out.println("Product ID already exists!");

        } else {

            inventory.put(product.productId, product);

            System.out.println("Product added.");

        }

    }

    // Update Product

    public void updateProduct(String productId, int quantity, double price) {

        Product p = inventory.get(productId);

        if (p != null) {

            p.quantity = quantity;

            p.price = price;

            System.out.println("Product updated.");

        } else {

            System.out.println("Product not found!");

        }

    }

    // Delete Product

    public void deleteProduct(String productId) {

        if (inventory.remove(productId) != null) {

            System.out.println("Product deleted.");

        } else {

            System.out.println("Product not found!");

        }

    }

    // Display Inventory

    public void displayInventory() {

        if (inventory.isEmpty()) {

            System.out.println("Inventory is empty.");

            return;

        }

        System.out.println("Inventory List:");

        for (Product p : inventory.values()) {

            p.display();

        }

    }

}

// Main Class

public class InventoryManagementSystem {

    public static void main(String[] args) {

        InventoryManager manager = new InventoryManager();

        Product p1 = new Product("P001", "Mouse", 100, 499.99);

        Product p2 = new Product("P002", "Keyboard", 50, 899.99);

        Product p3 = new Product("P003", "Monitor", 20, 7999.00);

        manager.addProduct(p1);

        manager.addProduct(p2);

        manager.addProduct(p3);

        manager.displayInventory();

        System.out.println("\nUpdating quantity and price for P002...");

        manager.updateProduct("P002", 70, 849.99);

        manager.displayInventory();

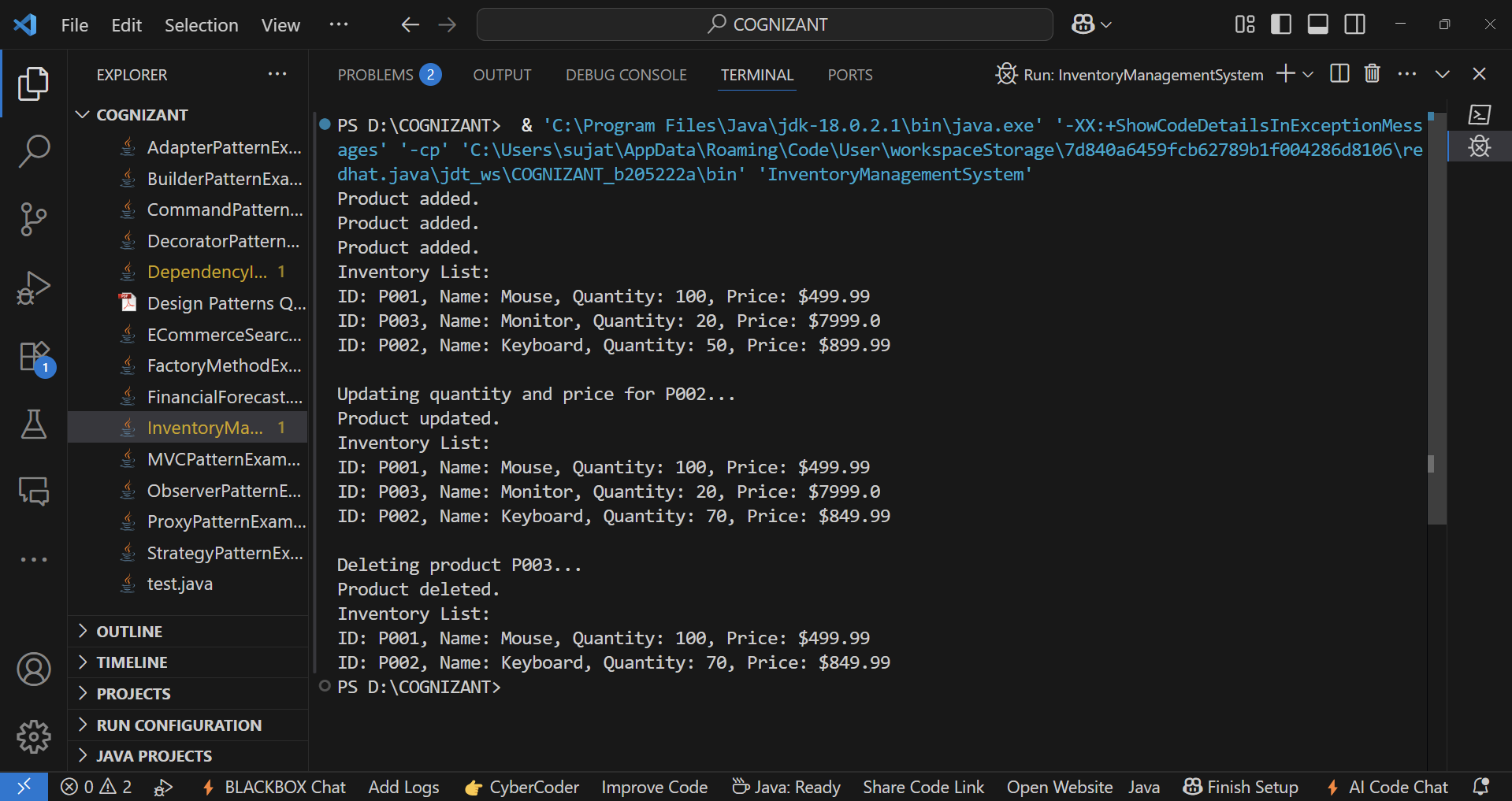
        System.out.println("\nDeleting product P003...");

        manager.deleteProduct("P003");

        manager.displayInventory();

    }

}

**OUTPUT:-**

**Step 4: Analysis**

Time Complexity with HashMap:

| **Operation** | **Time Complexity** | **Explanation** |
| --- | --- | --- |
| Add Product | O(1) avg | Direct hash-based insertion |
| Update Product | O(1) avg | Access via key |
| Delete Product | O(1) avg | Direct hash-based removal |
| Display All | O(n) | Iterates over all products |

**Optimizations:**

* Use TreeMap if you need sorted output by productId (O(log n) operations).
* Store frequently accessed data in cache (e.g., LRU) for even faster read times.