**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.**

**Solution:**

**InventoryManagementSystem.java**

import java.util.HashMap;

import java.util.Scanner;

class Product {

int productId;

String productName;

int quantity;

double price;

public Product(int productId, String productName, int quantity, double price) {

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

public String toString() {

return "ID: " + productId + ", Name: " + productName + ", Quantity: " + quantity + ", Price: ₹" + price;

}

}

public class InventoryManagementSystem {

static HashMap<Integer, Product> inventory = new HashMap<>();

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

while (true) {

System.out.println("\n1. Add Product");

System.out.println("2. Update Product");

System.out.println("3. Delete Product");

System.out.println("4. View Inventory");

System.out.println("5. View Time Complexity Analysis");

System.out.println("6. Exit");

System.out.print("Choose an option: ");

int choice = sc.nextInt();

switch (choice) {

case 1 -> addProduct();

case 2 -> updateProduct();

case 3 -> deleteProduct();

case 4 -> viewInventory();

case 5 -> viewAnalysis();

case 6 -> {

System.out.println("Exiting...");

return;

}

default -> System.out.println("Invalid choice!");

}

}

}

static void addProduct() {

System.out.print("Enter Product ID: ");

int id = sc.nextInt();

if (inventory.containsKey(id)) {

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

return;

}

sc.nextLine();

System.out.print("Enter Product Name: ");

String name = sc.nextLine();

System.out.print("Enter Quantity: ");

int qty = sc.nextInt();

System.out.print("Enter Price: ");

double price = sc.nextDouble();

Product p = new Product(id, name, qty, price);

inventory.put(id, p);

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

}

static void updateProduct() {

System.out.print("Enter Product ID to update: ");

int id = sc.nextInt();

if (!inventory.containsKey(id)) {

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

return;

}

sc.nextLine();

System.out.print("Enter New Product Name: ");

String name = sc.nextLine();

System.out.print("Enter New Quantity: ");

int qty = sc.nextInt();

System.out.print("Enter New Price: ");

double price = sc.nextDouble();

Product p = new Product(id, name, qty, price);

inventory.put(id, p);

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

}

static void deleteProduct() {

System.out.print("Enter Product ID to delete: ");

int id = sc.nextInt();

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

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

} else {

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

}

}

static void viewInventory() {

if (inventory.isEmpty()) {

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

return;

}

System.out.println("\n--- Inventory List ---");

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

System.out.println(p);

}

}

static void viewAnalysis() {

System.out.println("\n--- Time Complexity Analysis ---");

System.out.println("Data Structure Used: HashMap<Integer, Product>");

System.out.println("Operations and Time Complexity:");

System.out.println(" - Add Product -> O(1) average case");

System.out.println(" - Update Product -> O(1) average case");

System.out.println(" - Delete Product -> O(1) average case");

System.out.println(" - View Inventory -> O(n), where n = number of products");

System.out.println("\n--- Optimizations ---");

System.out.println("1. Use TreeMap instead of HashMap if sorted order is needed by Product ID.");

System.out.println("2. Use additional maps (e.g., by product name) for faster lookups by other attributes.");

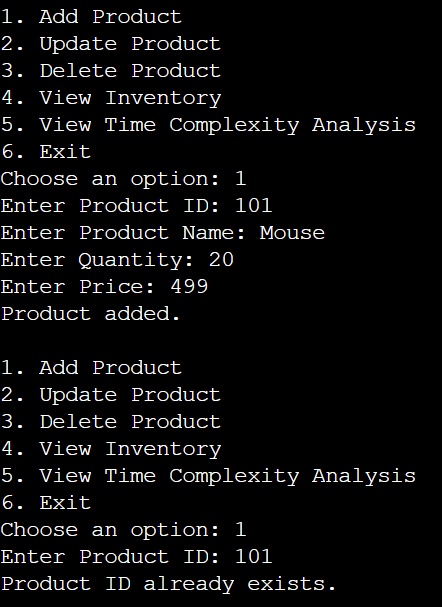
System.out.println("3. In real-world applications, use a database with indexing for large inventories.");

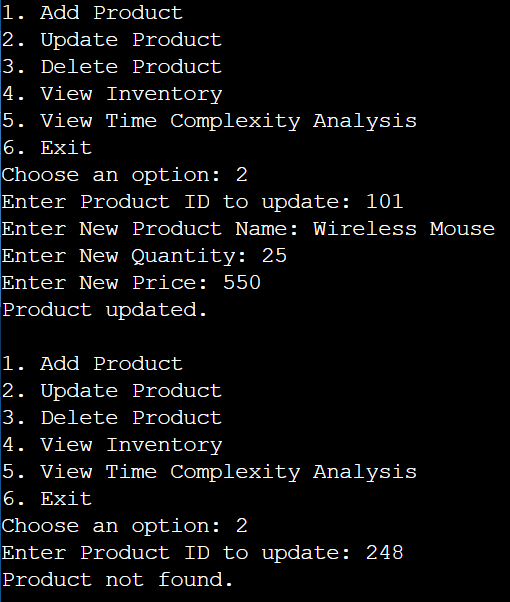
System.out.println("4. Consider caching frequently accessed products to reduce access time.");

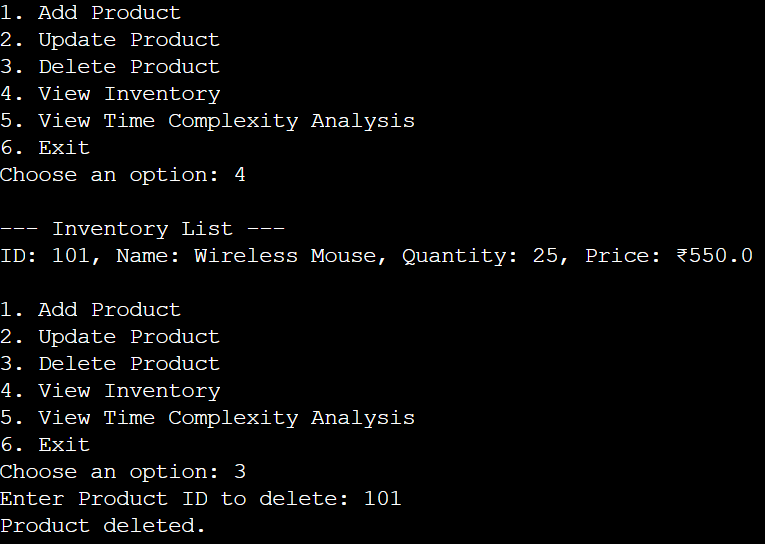
}

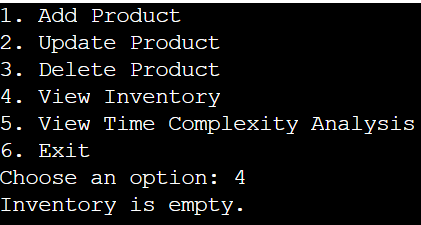
}

**Output:**

****

****

****

****

