**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.

import java.util.HashMap;

import java.util.Map;

class Product {

private String productId;

private String productName;

private int quantity;

private 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 getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public int getQuantity() {

return quantity;

}

public double getPrice() {

return price;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public void setPrice(double price) {

this.price = price;

}

public String toString() {

return "Product{" +

"productId='" + productId + '\'' +

", productName='" + productName + '\'' +

", quantity=" + quantity +

", price=" + price +

'}';

}

}

class InventoryManagementSystem {

private Map<String, Product> inventory;

public InventoryManagementSystem() {

inventory = new HashMap<>();

}

public void addProduct(Product product) {

inventory.put(product.getProductId(), product);

}

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

Product product = inventory.get(productId);

if (product != null) {

product.setQuantity(quantity);

product.setPrice(price);

} else {

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

}

}

public void deleteProduct(String productId) {

inventory.remove(productId);

}

public void displayProducts() {

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

System.***out***.println(product);

}

}

}

public class WeekOne {

public static void main(String[] args) {

InventoryManagementSystem ims = new InventoryManagementSystem();

ims.addProduct(new Product("P001", "Laptop", 5, 99999.99));

ims.addProduct(new Product("P002", "Smartphone", 15, 49999.99));

ims.addProduct(new Product("P003", "Tablet", 10, 79999.99));

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

ims.displayProducts();

ims.updateProduct("P002", 18, 45999.99);

System.***out***.println("\nAfter updating product P002:");

ims.displayProducts();

ims.deleteProduct("P001");

System.***out***.println("\nAfter deleting product P001:");

ims.displayProducts();

}

}

OUTPUT:

Inventory:

Product{productId='P001', productName='Laptop', quantity=5, price=99999.99}

Product{productId='P003', productName='Tablet', quantity=10, price=79999.99}

Product{productId='P002', productName='Smartphone', quantity=15, price=49999.99}

After updating product P002:

Product{productId='P001', productName='Laptop', quantity=5, price=99999.99}

Product{productId='P003', productName='Tablet', quantity=10, price=79999.99}

Product{productId='P002', productName='Smartphone', quantity=18, price=45999.99}

After deleting product P001:

Product{productId='P003', productName='Tablet', quantity=10, price=79999.99}

Product{productId='P002', productName='Smartphone', quantity=18, price=45999.99}

The program includes a Product class, uses a HashMap to store products, and implements methods to add, update, and delete products from the inventory.

1. Why Data Structures and Algorithms Are Essential in Handling Large Inventories

Data structures like HashMap are essential for efficient data storage and retrieval, especially when handling large inventories. HashMap allows for average O(1) time complexity for add, update, and delete operations, making it suitable for this scenario.

2. Discuss the types of data structures suitable for this problem.

Types of Data Structures Suitable for This Problem are:

ArrayList: Provides dynamic array capabilities, allowing for efficient random access and easy resizing.

HashMap: Offers constant time complexity for insertion, deletion, and look-up operations, making it ideal for inventory management.

TreeMap: Maintains sorted order of keys and allows for efficient range queries and ordered operations.

In this program, a HashMap is used to store products, where the key is the productId and the value is the Product object. This allows for quick access to products based on their IDs.

3. Class Definition:

The Product class has attributes for productId, productName, quantity, and price, along with appropriate getters, setters, and a toString method for displaying product details.

4. Methods:

The InventoryManagementSystem class contains methods to add, update, and delete products, as well as a method to display all products.

5. Time Complexity Analysis:

Add: O(1) on average (due to HashMap).

Update: O(1) on average (finding the product by ID).

Delete: O(1) on average (removing the product by ID).

6. Optimization:

-To optimize operations further, consider implementing features like batch updates or using a more complex data structure if the inventory grows significantly, such as a tree structure for sorted access.