**Exercise 1: Inventory Management System**

**Q) Explain why data structures and algorithms are essential in handling large inventories.**

Data structures help store and organize data efficiently, while algorithms enable optimized operations on that data. In the context of large inventories, it’s crucial to perform tasks like searching for a product by its ID, adding or removing products, and updating product details—quickly and accurately.

Without proper data structures (like HashMap, ArrayList, or TreeMap) and efficient algorithms (like binary search or sorting techniques), these operations can become slow and inefficient as the size of the inventory grows. Therefore, using the right combination of data structures and algorithms is essential to ensure fast access, scalability, and overall system performance when managing large inventories.

**Q) Discuss the types of data structures suitable for this problem.**

The suitable data structures for handling inventory management include HashMap, ArrayList, and TreeMap.

* ArrayList is simple to use but inefficient for search, update, or delete operations, as it requires linear time (O(n)).
* TreeMap maintains the elements in sorted order and supports logarithmic time complexity (O(log n)) for most operations, which is useful if sorting is needed.
* HashMap is the most efficient among the three for this problem. It provides constant-time (O(1)) performance for operations like add, delete, update, and search (on average), when accessed using the product ID as the key.

Hence, HashMap is ideal for managing large inventories where quick access and modification are essential.

**Code:**

**Product Class: -**

public class Product {  
 private String productId;  
 private String productName;  
 private Integer quantity;  
 private double price;  
  
 public Product(String productId, String productName, Integer quantity, double price) {  
 this.productId = productId;  
 this.productName = productName;  
 this.quantity = quantity;  
 this.price = price;  
 }  
  
 public String getProductId() {  
 return productId;  
 }  
 public void setProductId(String productId) {  
 this.productId = productId;  
 }  
 public String getProductName() {  
 return productName;  
 }  
 public void setProductName(String productName) {  
 this.productName = productName;  
 }  
 public Integer getQuantity() {  
 return quantity;  
 }  
 public void setQuantity(Integer quantity) {  
 this.quantity = quantity;  
 }  
 public double getPrice() {  
 return price;  
 }  
 public void setPrice(double price) {  
 this.price = price;  
 }  
}

**InventoryManager Class: -**

import java.util.HashMap;  
  
public class InventoryManager {  
 private HashMap<String,Product> inventory;  
 public InventoryManager(){  
 inventory=new HashMap<>();  
 }  
 public void addProduct(Product product){  
 if (inventory.containsKey(product.getProductId())){  
 System.*out*.println("Product already exists.");  
 } else {  
 inventory.put(product.getProductId(),product);  
 System.*out*.println("Product added successfully.");  
 }  
 }  
  
 public void updateProduct(String productId,String name,int quantity,double price){  
 if (inventory.containsKey(productId)){  
 Product p=inventory.get(productId);  
 p.setProductName(name);  
 p.setQuantity(quantity);  
 p.setPrice(price);  
 System.*out*.println("Product updated.");  
 }else{  
 System.*out*.println("Product not found.");  
 }  
 }  
  
 public void deleteProduct(String productId){  
 if (inventory.remove(productId) != null){  
 System.*out*.println("Product deleted.");  
 }else{  
 System.*out*.println("Product not found.");  
 }  
 }  
}

**Main Class: -**

public class Main {  
 public static void main(String[] args){  
 InventoryManager manager=new InventoryManager();  
 Product p1=new Product("p\_01","Watch",20,599.00);  
 Product p2=new Product("p\_02","Iphone 15",50,61990.00);  
 Product p3=new Product("p\_03","Laptop",10,59000.00);  
 manager.addProduct(p1);  
 manager.addProduct(p2);  
 manager.addProduct(p3);  
 manager.updateProduct("p\_02","Mouse",15,699.00);  
 manager.deleteProduct(p3.getProductId());  
 manager.updateProduct("p\_03","Keyboard",10,1099.00);  
 }  
}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Q) Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.Discuss how you can optimize these operations.**

1.Add Product: The time complexity for adding a product to a HashMap is O(1) on average. This is because the product is inserted based on its unique key (productId). In very few cases where the internal array needs to be resized (rehashing), the time complexity can go up to O(n) temporarily.

2.Update Product: Updating a product is also O(1) since the HashMap allows direct access to the product using its key. Once the product is retrieved, its fields can be modified without needing to search through the entire data set.

3.Delete Product: Deletion is O(1) on average as well, because the HashMap removes the product based on its key without the need for iteration.

4.List Products: Listing all products involves traversing all values in the map, which takes O(n) time, where *n* is the number of products in the inventory.

5.Searching for a product by its ID in a HashMap is extremely efficient, with a time complexity of O(1) on average. The product can be retrieved instantly using its key without scanning the entire structure.