**Exercise 1: Inventory Management System**

\* Why Data Structures?

- Efficient storage and retrieval are vital for large inventories.

- Choosing the right data structure (like HashMap) ensures fast access, updates, and deletions.

\* Suitable Data Structures:

- ArrayList: Good for ordered data, but search/delete is O(n).

- HashMap: Offers average-case O(1) time for add, update, delete, and search.

import java.util.\*;

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;

}

@Override

public String toString() {

return "[" + productId + "] " + productName + " - Qty: " + quantity + ", Price: Rs." + price;

}

}

public class Main {

private Map<Integer, Product> inventory;

public Main() {

inventory = new HashMap<>();

}

// Add product to inventory

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: " + product.productName);

}

}

// Update product information

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

if (!inventory.containsKey(productId)) {

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

} else {

Product p = inventory.get(productId);

p.productName = name;

p.quantity = qty;

p.price = price;

System.out.println("Product updated: " + productId);

}

}

// Delete a product from inventory

public void deleteProduct(int productId) {

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

System.out.println("Product deleted: " + productId);

} else {

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

}

}

// Display all products in inventory

public void displayInventory() {

if (inventory.isEmpty()) {

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

} else {

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

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

System.out.println(p);

}

}

}

// Main method to test the system

public static void main(String[] args) {

Main ims = new Main();

ims.addProduct(new Product(101, "Laptop", 10, 999.99));

ims.addProduct(new Product(102, "Mouse", 50, 25.99));

ims.addProduct(new Product(103, "Keyboard", 30, 45.49));

System.out.println();

ims.displayInventory();

System.out.println("\nUpdating product 102...");

ims.updateProduct(102, "Wireless Mouse", 60, 29.99);

ims.displayInventory();

System.out.println("\nDeleting product 101...");

ims.deleteProduct(101);

ims.displayInventory();

}}

Time Complexity (HashMap):

\* Add -> O(1) average

\* Update -> O(1) average

\* Delete -> O(1) average

\* Lookup -> O(1) average

\* Worst case (due to collisions): O(n)

\* Optimization:

- Choose good hash functions.

- Use TreeMap if you need sorted keys.

- For massive datasets, consider database + indexing.

