**Exercise 1: Inventory Management System**

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

**1. Understand the Problem:**

**Why are data structures and algorithms essential in handling large inventories?**

* In a warehouse, there can be thousands of products. Efficient access, updates, and deletions (CRUD operations) are necessary to maintain performance.
* Choosing the right data structure improves speed and memory usage.
* Algorithms define how efficiently tasks like search, sort, or update operations are performed.

**Suitable Data Structures:**

* **HashMap**: For fast retrieval and update of products based on unique productId.
* **ArrayList**: Suitable for ordered data, but less efficient for search/update/delete.
* **TreeMap**: Keeps data sorted by keys; useful for range queries.

**2. Setup:**

* Create a Java project named InventoryManagementSystem.
* Create three Java classes:
  + Product.java
  + Inventory.java
  + Main.java

**3. Implementation:**

**Product.java**

public 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 "Product [ID=" + productId + ", Name=" + productName + ", Qty=" + quantity + ", Price=" + price + "]";

}

}

**Inventory.java**

import java.util.\*;

public class Inventory {

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

public void addProduct(Product product) {

inventory.put(product.productId, product);

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

}

public void updateProduct(Product product) {

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

inventory.put(product.productId, product);

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

} else {

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

}

}

public void deleteProduct(int productId) {

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

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

} else {

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

}

}

public Product getProduct(int productId) {

return inventory.get(productId);

}

public void displayInventory() {

if (inventory.isEmpty()) {

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

} else {

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

System.out.println(p);

}

}

}

}

**Main.java**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Inventory inventory = new Inventory();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("\n=== Inventory Management System ===");

System.out.println("1. Add Product");

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

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

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

System.out.println("5. Search Product");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

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

int id = scanner.nextInt();

scanner.nextLine();

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

String name = scanner.nextLine();

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

int qty = scanner.nextInt();

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

double price = scanner.nextDouble();

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

inventory.addProduct(p);

break;

case 2:

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

id = scanner.nextInt();

scanner.nextLine();

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

name = scanner.nextLine();

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

qty = scanner.nextInt();

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

price = scanner.nextDouble();

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

inventory.updateProduct(p);

break;

case 3:

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

id = scanner.nextInt();

inventory.deleteProduct(id);

break;

case 4:

inventory.displayInventory();

break;

case 5:

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

id = scanner.nextInt();

Product product = inventory.getProduct(id);

System.out.println(product != null ? product : "Product not found.");

break;

case 0:

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

break;

default:

System.out.println("Invalid choice.");

}

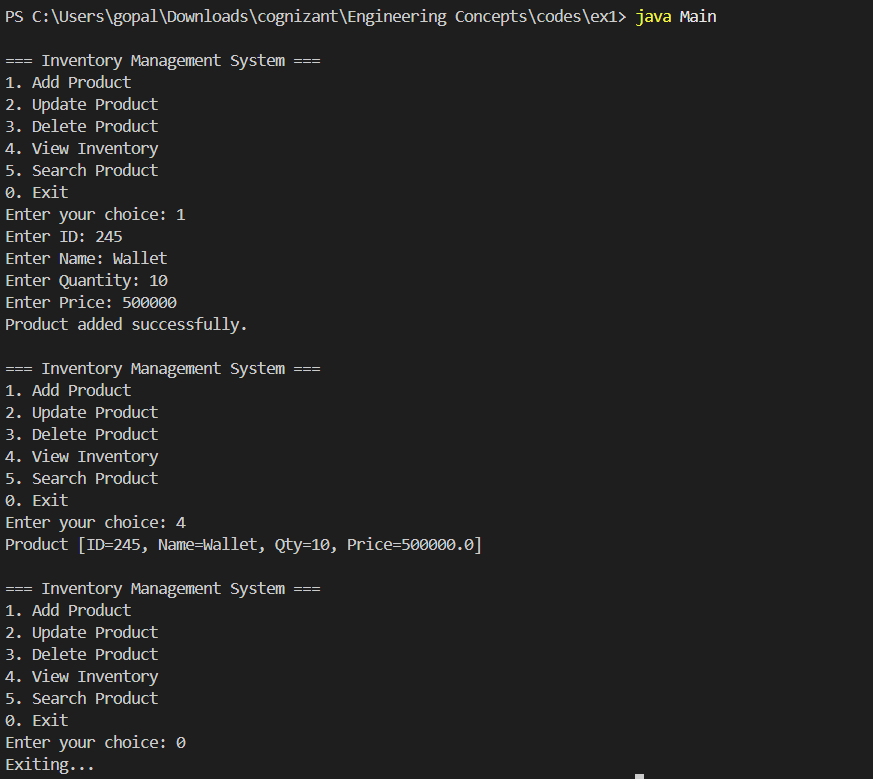
} while (choice != 0);

scanner.close();

}

}

**Output:**



**4. Analysis:**

**Time Complexity (using HashMap):**

* Add Product: **O(1)** average case
* Update Product: **O(1)** average case
* Delete Product: **O(1)** average case
* Search Product: **O(1)** average case
* Display Inventory: **O(n)** where *n* is the number of products

**Optimizations:**

* Use TreeMap if sorted product listing is required.
* Use indexing strategies if complex queries are needed.
* For concurrent environments, use ConcurrentHashMap.
* Add indexing or separate maps for attributes like productName for faster custom search.

**Conclusion:**  
By choosing an efficient data structure like HashMap, we ensure fast operations crucial for large inventories. The implementation supports all basic CRUD operations and is scalable for future features like file/database storage or product filtering.