// 1. Understanding the Problem:

// Data structures and algorithms are critical for inventory management because:

// Large inventories require efficient storage and retrieval to handle thousands or millions of products.

// Fast operations (add, update, delete) are necessary for real-time inventory tracking.

// Scalability is essential to accommodate growing inventory sizes without performance degradation.

// Suitable data structures:

// HashMap provides O(1) average-case time complexity for add, update, and delete operations using productId as the key.

// ArrayList is Suitable for sequential access but O(n) for search and delete, less ideal for large inventories.

// TreeMap is Useful for sorted operations but O(log n) for basic operations, slower than HashMap.

// HashMap is chosen for its O(1) average-case performance for key-based operations, ideal for productId-based lookups.

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

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 setProductName(String productName) { this.productName = productName; }

    public void setQuantity(int quantity) { this.quantity = quantity; }

    public void setPrice(double price) { this.price = price; }

    @Override

    public String toString() {

        return "Product{ID=" + productId + ", Name=" + productName +

               ", Quantity=" + quantity + ", Price=Rs" + price + "}";

    }

}

class InventoryManagementSystem {

    private Map<String, Product> inventory;

    public InventoryManagementSystem() {

        inventory = new HashMap<>();

    }

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

        if (inventory.containsKey(productId)) {

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

            return;

        }

        Product product = new Product(productId, productName, quantity, price);

        inventory.put(productId, product);

        System.out.println("Product added: " + product);

    }

    public void updateProduct(String productId, String productName, Integer quantity, Double price) {

        Product product = inventory.get(productId);

        if (product == null) {

            System.out.println("Product with ID " + productId + " not found.");

            return;

        }

        if (productName != null && !productName.isEmpty()) product.setProductName(productName);

        if (quantity != null) product.setQuantity(quantity);

        if (price != null) product.setPrice(price);

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

    }

    public void deleteProduct(String productId) {

        Product removed = inventory.remove(productId);

        if (removed == null) {

            System.out.println("Product with ID " + productId + " not found.");

            return;

        }

        System.out.println("Product removed: " + removed);

    }

    public void getProduct(String productId) {

        Product product = inventory.get(productId);

        if (product == null) {

            System.out.println("Product with ID " + productId + " not found.");

            return;

        }

        System.out.println("Retrieved: " + product);

    }

}

public class Main {

    public static void main(String[] args) {

        InventoryManagementSystem ims = new InventoryManagementSystem();

        Scanner scanner = new Scanner(System.in);

        boolean running = true;

        while (running) {

            System.out.println("\nInventory Management System Menu:");

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

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

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

            System.out.println("4. Retrieve Product");

            System.out.println("5. Exit");

            System.out.print("Enter your choice (1-5): ");

            int choice;

            try {

                choice = Integer.parseInt(scanner.nextLine());

            } catch (NumberFormatException e) {

                System.out.println("Invalid input. Please enter a number between 1 and 5.");

                continue;

            }

            switch (choice) {

                case 1: // Add Product

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

                    String addId = scanner.nextLine();

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

                    String addName = scanner.nextLine();

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

                    int addQuantity;

                    try {

                        addQuantity = Integer.parseInt(scanner.nextLine());

                    } catch (NumberFormatException e) {

                        System.out.println("Invalid quantity. Operation cancelled.");

                        break;

                    }

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

                    double addPrice;

                    try {

                        addPrice = Double.parseDouble(scanner.nextLine());

                    } catch (NumberFormatException e) {

                        System.out.println("Invalid price. Operation cancelled.");

                        break;

                    }

                    ims.addProduct(addId, addName, addQuantity, addPrice);

                    break;

                case 2: // Update Product

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

                    String updateId = scanner.nextLine();

                    System.out.print("Enter new Product Name (or press Enter to skip): ");

                    String updateName = scanner.nextLine();

                    System.out.print("Enter new Quantity (or press Enter to skip): ");

                    Integer updateQuantity = null;

                    String quantityInput = scanner.nextLine();

                    if (!quantityInput.isEmpty()) {

                        try {

                            updateQuantity = Integer.parseInt(quantityInput);

                        } catch (NumberFormatException e) {

                            System.out.println("Invalid quantity. Operation cancelled.");

                            break;

                        }

                    }

                    System.out.print("Enter new Price (or press Enter to skip): ");

                    Double updatePrice = null;

                    String priceInput = scanner.nextLine();

                    if (!priceInput.isEmpty()) {

                        try {

                            updatePrice = Double.parseDouble(priceInput);

                        } catch (NumberFormatException e) {

                            System.out.println("Invalid price. Operation cancelled.");

                            break;

                        }

                    }

                    ims.updateProduct(updateId, updateName, updateQuantity, updatePrice);

                    break;

                case 3:

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

                    String deleteId = scanner.nextLine();

                    ims.deleteProduct(deleteId);

                    break;

                case 4: // Retrieve Product

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

                    String getId = scanner.nextLine();

                    ims.getProduct(getId);

                    break;

                case 5: // Exit

                    running = false;

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

                    break;

                default:

                    System.out.println("Invalid choice. Please enter a number between 1 and 5.");

            }

        }

        scanner.close();

    }

}

// Analysis (unchanged from previous version):

// Time Complexity (HashMap-based implementation):

// - Add (put): O(1) average case

// - Update (get + set): O(1) average case

// - Delete (remove): O(1) average case

// - Retrieve (get): O(1) average case

// Optimization strategies and limitations remain the same as previously described.

// 4. Analysis:

// Time Complexity (HashMap-based implementation):

// - Add (put): O(1) average case, as HashMap uses hashing for key-based storage.

// - Update (get + set): O(1) average case for retrieval and updating fields.

// - Delete (remove): O(1) average case for key-based removal.

// - Retrieve (get): O(1) average case for key-based lookup.

// Optimization strategies:

// - Implement caching for frequently accessed products to reduce lookup time.

// - Use a database (e.g., SQL) for persistent storage and large-scale data, with HashMap as an in-memory cache.

// - Add indexing on productName for faster searches by name, using a secondary HashMap or TreeMap.

// Limitations:

// - HashMap has O(n) worst-case complexity due to hash collisions, mitigated by good hash functions.

// - Memory usage grows linearly with the number of products, which could be optimized with a database for large inventories.

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A black screen with white text

AI-generated content may be incorrect.**