Exercise 1

Inventory management System

import java.util.HashMap;

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 productId + " - " + productName + " (" + quantity + ") - Rs. " + price;

}

}

public class InventoryManager {

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

public void addProduct(Product p) {

inventory.put(p.productId, p);

}

public void updateProduct(Product p) {

inventory.put(p.productId, p);

}

public void deleteProduct(int productId) {

inventory.remove(productId);

}

public Product getProduct(int productId) {

return inventory.get(productId);

}

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

Product p1 = new Product(1, "Mouse", 10, 299.99);

Product p2 = new Product(2, "Keyboard", 5, 749.50);

manager.addProduct(p1);

manager.addProduct(p2);

System.out.println(manager.getProduct(1));

manager.updateProduct(new Product(1, "Mouse", 15, 299.99));

System.out.println(manager.getProduct(1));

manager.deleteProduct(2);

System.out.println(manager.getProduct(2));

}

}

Exercise 2

Ecommerce Search

import java.util.Arrays;

class Product {

int id;

String productName;

public Product(int id, String name) {

this.id = id;

this.productName = name;

}

public String toString() {

return id + ": " + productName;

}

}

public class SearchProduct {

public Product linearSearch(Product[] products, String name) {

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) return p;

}

return null;

}

public Product binarySearch(Product[] products, String name) {

int low = 0, high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = name.compareToIgnoreCase(products[mid].productName);

if (cmp == 0) return products[mid];

else if (cmp < 0) high = mid - 1;

else low = mid + 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Book"),

new Product(2, "Laptop"),

new Product(3, "Mouse")

};

Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName)); // required for binary search

SearchProduct sp = new SearchProduct();

System.out.println(sp.linearSearch(products, "Mouse"));

System.out.println(sp.binarySearch(products, "Laptop"));

}

}

Exercise 3

Sorting customer orders

class Order {

int orderId;

double totalPrice;

public Order(int id, double price) {

this.orderId = id;

this.totalPrice = price;

}

public String toString() {

return "Order " + orderId + " - Rs. " + totalPrice;

}

}

public class OrderSorter {

public void bubbleSort(Order[] orders) {

for (int i = 0; i < orders.length - 1; i++) {

for (int j = 0; j < orders.length - i - 1; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

}

}

}

}

public static void main(String[] args) {

Order[] orders = {

new Order(1, 300.0),

new Order(2, 100.0),

new Order(3, 200.0)

};

OrderSorter sorter = new OrderSorter();

sorter.bubbleSort(orders);

for (Order o : orders) {

System.out.println(o);

}

}

}

Exercise 4

Employee Management Systems

class Employee {

int employeeId;

String name;

public Employee(int id, String name) {

this.employeeId = id;

this.name = name;

}

public String toString() {

return employeeId + " - " + name;

}

}

public class EmployeeManager {

Employee[] employees = new Employee[100];

int count = 0;

public void addEmployee(Employee e) {

employees[count++] = e;

}

public Employee searchEmployee(int id) {

for (int i = 0; i < count; i++) {

if (employees[i].employeeId == id) return employees[i];

}

return null;

}

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager();

manager.addEmployee(new Employee(101, "Alice"));

manager.addEmployee(new Employee(102, "Bob"));

System.out.println(manager.searchEmployee(101));

System.out.println(manager.searchEmployee(999)); // not found

}

}

Exercise 5

Task Management using Linked List

class Task {

String taskName;

public Task(String name) {

this.taskName = name;

}

}

class TaskNode {

Task task;

TaskNode next;

}

public class TaskList {

TaskNode head;

public void addTask(Task task) {

TaskNode newNode = new TaskNode();

newNode.task = task;

newNode.next = head;

head = newNode;

}

public void printTasks() {

TaskNode current = head;

while (current != null) {

System.out.println(current.task.taskName);

current = current.next;

}

}

public static void main(String[] args) {

TaskList tasks = new TaskList();

tasks.addTask(new Task("Design UI"));

tasks.addTask(new Task("Build Backend"));

tasks.addTask(new Task("Write Tests"));

tasks.printTasks();

}

}

Exercise 6

Library Management System

class Book {

String title;

public Book(String title) {

this.title = title;

}

public String toString() {

return title;

}

}

public class LibraryManager {

public Book linearSearch(Book[] books, String title) {

for (Book b : books) {

if (b.title.equalsIgnoreCase(title)) return b;

}

return null;

}

public static void main(String[] args) {

Book[] books = {

new Book("C Programming"),

new Book("Java Basics"),

new Book("Python 101")

};

LibraryManager lm = new LibraryManager();

System.out.println(lm.linearSearch(books, "Java Basics"));

System.out.println(lm.linearSearch(books, "C++")); // not found

}

}

Exercise 7

Weather Forecasting

public class FinancialForecast {

public double predictValue(double current, double growthRate, int years) {

if (years == 0) return current;

return predictValue(current \* (1 + growthRate), growthRate, years - 1);

}

public static void main(String[] args) {

FinancialForecast forecast = new FinancialForecast();

double futureValue = forecast.predictValue(10000, 0.10, 3); // 10% growth for 3 years

System.out.println("Future Value: Rs. " + futureValue);

}

}