**Exercise 1: Inventory Management System**

**Implementation:**

import java.util.HashMap;

import java.util.Scanner;

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 void display() {

System.out.println("ID: " + productId + ", Name: " + productName + ", Quantity: " + quantity + ", Price: " + price);

}

}

public class Main {

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

static Scanner sc = new Scanner(System.in);

public static void addProduct() {

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

int id = sc.nextInt();

sc.nextLine(); // Consume newline

if (inventory.containsKey(id)) {

System.out.println("Product ID already exists!");

return;

}

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

String name = sc.nextLine();

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

int quantity = sc.nextInt();

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

double price = sc.nextDouble();

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

inventory.put(id, p);

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

}

public static void updateProduct() {

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

int id = sc.nextInt();

if (!inventory.containsKey(id)) {

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

return;

}

Product p = inventory.get(id);

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

p.quantity = sc.nextInt();

System.out.print("Enter new price: ");

p.price = sc.nextDouble();

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

}

public static void deleteProduct() {

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

int id = sc.nextInt();

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

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

} else {

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

}

}

public static void displayInventory() {

if (inventory.isEmpty()) {

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

return;

}

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

p.display();

}

}

public static void main(String[] args) {

int choice;

do {

System.out.println("\n--- Inventory Menu ---");

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

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

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

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

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

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

choice = sc.nextInt();

switch (choice) {

case 1: addProduct();

break;

case 2: updateProduct();

break;

case 3: deleteProduct();

break;

case 4: displayInventory();

break;

case 5: System.out.println("Exiting...");

break;

default: System.out.println("Invalid choice!");

}

} while (choice != 5);

}

}

**OUTPUT:**

**//I have given different choices to ensure that my code performs perfectly**

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 1

Enter Product ID: 101

Enter Product Name: TV

Enter Quantity: 4

Enter Price: 25000

Product added.

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 1

Enter Product ID: 102

Enter Product Name: AirConditioner

Enter Quantity: 8

Enter Price: 35000

Product added.

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 1

Enter Product ID: 103

Enter Product Name: cooler

Enter Quantity: 2

Enter Price: 15000

Product added.

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 4

ID: 101, Name: TV, Quantity: 4, Price: 25000.0

ID: 102, Name: AirConditioner, Quantity: 8, Price: 35000.0

ID: 103, Name: cooler, Quantity: 2, Price: 15000.0

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 2

Enter Product ID to update: 101

Enter New Quantity: 8

Enter new price: 35000

Product updated successfully.

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 4

ID: 101, Name: TV, Quantity: 8, Price: 35000.0

ID: 102, Name: AirConditioner, Quantity: 8, Price: 35000.0

ID: 103, Name: cooler, Quantity: 2, Price: 15000.0

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 3

Enter Product ID to delete: 103

Product deleted successfully.

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 4

ID: 101, Name: TV, Quantity: 8, Price: 35000.0

ID: 102, Name: AirConditioner, Quantity: 8, Price: 35000.0

--- Inventory Menu ---

1. Add Product

2. Update Product

3. Delete Product

4. Display Inventory

5. Exit

Enter choice: 5

Exiting...

**Exercise 2: E-commerce Platform Search Function**

**Implementation:**

package com.ecommerce.search;

public class Product {

int productId;

String productName;

String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public void display() {

System.out.println("ID: " + productId + ", Name: " + productName + ", Category: " + category);

}

}

// ECommerceSearch.java Class (Main class)

package com.ecommerce.search;

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class ECommerceSearch {

static Product[] products = {

new Product(104, "Keyboard", "Electronics"),

new Product(101, "Laptop", "Electronics"),

new Product(102, "Shoes", "Fashion"),

new Product(103, "Book", "Education")

};

public static Product linearSearch(String name) {

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

public static Product binarySearch(int id) {

Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (products[mid].productId == id) {

return products[mid];

} else if (products[mid].productId < id) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- E-commerce Search Menu ---");

System.out.println("1. Search by Product Name (Linear Search)");

System.out.println("2. Search by Product ID (Binary Search)");

System.out.println("3. Display All Products");

System.out.println("4. Exit");

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

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

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

String name = sc.nextLine();

Product result1 = linearSearch(name);

if (result1 != null) result1.display();

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

break;

case 2:

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

int id = sc.nextInt();

Product result2 = binarySearch(id);

if (result2 != null) result2.display();

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

break;

case 3:

for (Product p : products) {

p.display();

}

break;

case 4:

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

break;

default:

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

}

} while (choice != 4);

}

}

**OUTPUT:**

--- E-commerce Search Menu ---

1. Search by Product Name (Linear Search)

2. Search by Product ID (Binary Search)

3. Display All Products

4. Exit

Enter your choice: 1

Enter Product Name to Search: TV

ID: 101, Name: TV, Category: Electronics

--- E-commerce Search Menu ---

1. Search by Product Name (Linear Search)

2. Search by Product ID (Binary Search)

3. Display All Products

4. Exit

Enter your choice: 2

Enter Product ID to Search: 101

ID: 101, Name: TV, Category: Electronics

--- E-commerce Search Menu ---

1. Search by Product Name (Linear Search)

2. Search by Product ID (Binary Search)

3. Display All Products

4. Exit

Enter your choice: 3

ID: 101, Name: TV, Category: Electronics

ID: 102, Name: Shoes, Category: Fashion

ID: 103, Name: Book, Category: Education

ID: 104, Name: AC, Category: Electronics

--- E-commerce Search Menu ---

1. Search by Product Name (Linear Search)

2. Search by Product ID (Binary Search)

3. Display All Products

4. Exit

Enter your choice: 4

Exiting...

**Exercise 3: Sorting Customer Orders**

**Implementation:**

class Order {

int orderId;

String customerName;

double totalPrice;

public Order(int orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public void display() {

System.out.println("OrderID: " + orderId + ", Name: " + customerName + ", Total Price: " + totalPrice);

}

}

//sorting

import java.util.Scanner;

public class OrderSorting {

static Order[] orders = {

new Order(101, "shru", 6500.00),

new Order(102, "dev", 1200.50),

new Order(103, "harsh", 5800.75),

new Order(104, "kavita, 3200.00),

new Order(105, "amrita", 2200.25)

};

public static void bubbleSort(Order[] arr) {

int n = arr.length;

boolean swapped;

for (int i = 0; i < n - 1; i++) {

swapped = false;

for (int j = 0; j < n - i - 1; j++) {

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

Order temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swapped = true;

}

}

if (!swapped)

break;

}

}

public static void quickSort(Order[] arr, int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

public static int partition(Order[] arr, int low, int high) {

double pivot = arr[high].totalPrice;

int i = (low - 1);

for (int j = low; j < high; j++) {

if (arr[j].totalPrice <= pivot) {

i++;

Order temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

Order temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

public static void displayOrders(Order[] arr) {

for (Order o : arr) {

o.display();

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

Order[] copy;

do {

System.out.println("\n--- Sort Orders by Total Price ---");

System.out.println("1. Display Original Orders");

System.out.println("2. Sort using Bubble Sort");

System.out.println("3. Sort using Quick Sort");

System.out.println("4. Exit");

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

choice = sc.nextInt();

switch (choice) {

case 1:

System.out.println("\nOriginal Orders:");

displayOrders(orders);

break;

case 2:

copy = orders.clone();

bubbleSort(copy);

System.out.println("\nOrders Sorted using Bubble Sort:");

displayOrders(copy);

break;

case 3:

copy = orders.clone();

quickSort(copy, 0, copy.length - 1);

System.out.println("\nOrders Sorted using Quick Sort:");

displayOrders(copy);

break;

case 4:

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

break;

default:

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

}

} while (choice != 4);

}

}

**Output:**

--- Sort Orders by Total Price ---

1. Display Original Orders

2. Sort using Bubble Sort

3. Sort using Quick Sort

4. Exit

Enter your choice: 1

Original Orders:

OrderID: 101, Name: Shru, Total Price: 6500.0

OrderID: 102, Name: Dev, Total Price: 2200.5

OrderID: 103, Name: Harsh, Total Price: 5800.75

OrderID: 104, Name: Kavita, Total Price: 3200.0

OrderID: 105, Name: Amrita, Total Price: 2200.25

--- Sort Orders by Total Price ---

1. Display Original Orders

2. Sort using Bubble Sort

3. Sort using Quick Sort

4. Exit

Enter your choice: 2

Orders Sorted using Bubble Sort:

OrderID: 105, Name: Amrita, Total Price: 2200.25

OrderID: 102, Name: Dev, Total Price: 2200.5

OrderID: 104, Name: Kavita, Total Price: 3200.0

OrderID: 103, Name: Harsh, Total Price: 5800.75

OrderID: 101, Name: Shru, Total Price: 6500.0

--- Sort Orders by Total Price ---

1. Display Original Orders

2. Sort using Bubble Sort

3. Sort using Quick Sort

4. Exit

Enter your choice: 3

Orders Sorted using Quick Sort:

OrderID: 105, Name: Amrita, Total Price: 2200.25

OrderID: 102, Name: Dev, Total Price: 2200.5

OrderID: 104, Name: Kavita, Total Price: 3200.0

OrderID: 103, Name: Harsh, Total Price: 5800.75

OrderID: 101, Name: Shru, Total Price: 6500.0

--- Sort Orders by Total Price ---

1. Display Original Orders

2. Sort using Bubble Sort

3. Sort using Quick Sort

4. Exit

Enter your choice: 4

Exiting...

**Exercise 4: Employee Management System**

**Implementation:**

class Employee {

int employeeId;

String name;

String position;

double salary;

public Employee(int employeeId, String name, String position, double salary) {

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

public void display() {

System.out.println("ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: " + salary);

}

}

//arrays

import java.util.Scanner;

public class EmployeeManagementSystem {

static Employee[] employees = new Employee[100]; // Fixed size array

static int count = 0;

public static void addEmployee(Employee e) {

if (count < employees.length) {

employees[count++] = e;

System.out.println("Employee added.");

} else {

System.out.println("Employee list is full.");

}

}

public static Employee searchEmployee(int id) {

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

if (employees[i].employeeId == id) {

return employees[i];

}

}

return null;

}

public static void traverseEmployees() {

if (count == 0) {

System.out.println("No employees found.");

return;

}

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

employees[i].display();

}

}

public static boolean deleteEmployee(int id) {

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

if (employees[i].employeeId == id) {

for (int j = i; j < count - 1; j++) {

employees[j] = employees[j + 1];

}

employees[--count] = null; // Clear last

return true;

}

}

return false;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Employee Management Menu ---");

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

System.out.println("2. Search Employee by ID");

System.out.println("3. Display All Employees");

System.out.println("4. Delete Employee by ID");

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

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

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

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

int id = sc.nextInt();

sc.nextLine();

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

String name = sc.nextLine();

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

String position = sc.nextLine();

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

double salary = sc.nextDouble();

Employee emp = new Employee(id, name, position, salary);

addEmployee(emp);

break;

case 2:

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

int searchId = sc.nextInt();

Employee found = searchEmployee(searchId);

if (found != null) found.display();

else System.out.println("Employee not found.");

break;

case 3:

traverseEmployees();

break;

case 4:

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

int delId = sc.nextInt();

boolean success = deleteEmployee(delId);

System.out.println(success ? "Deleted successfully." : "Employee not found.");

break;

case 5:

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

break;

default:

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

}

} while (choice != 5);

}

}

**Output:**

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 1

Enter ID: 101

Enter Name: Shru

Enter Position: Senior Manager

Enter Salary: 95000

Employee added.

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 102

Invalid choice!

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 1

Enter ID: 102

Enter Name: Dev

Enter Position: Junior Analyst

Enter Salary: 75000

Employee added.

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 2

Enter ID to search: 101

ID: 101, Name: Shru, Position: Senior Manager, Salary: 95000.0

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 3

ID: 101, Name: Shru, Position: Senior Manager, Salary: 95000.0

ID: 102, Name: Dev, Position: Junior Analyst, Salary: 75000.0

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 4

Enter ID to delete: 102

Deleted successfully.

--- Employee Management Menu ---

1. Add Employee

2. Search Employee by ID

3. Display All Employees

4. Delete Employee by ID

5. Exit

Enter choice: 5

Exiting...

**Exercise 5: Task Management System**

**Implementation:**

class Task {

int taskId;

String taskName;

String status;

Task next;

public Task(int taskId, String taskName, String status) {

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

public void display() {

System.out.println("TaskID: " + taskId + ", Name: " + taskName + ", Status: " + status);

}

}

//singly linked list

import java.util.Scanner;

public class TaskManagementSystem {

Task head = null;

public void addTask(int id, String name, String status) {

Task newTask = new Task(id, name, status);

if (head == null) {

head = newTask;

} else {

Task temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newTask;

}

System.out.println("Task added.");

}

public Task searchTask(int id) {

Task temp = head;

while (temp != null) {

if (temp.taskId == id)

return temp;

temp = temp.next;

}

return null;

}

public void traverseTasks() {

if (head == null) {

System.out.println("No tasks available.");

return;

}

Task temp = head;

while (temp != null) {

temp.display();

temp = temp.next;

}

}

public boolean deleteTask(int id) {

if (head == null) return false;

if (head.taskId == id) {

head = head.next;

return true;

}

Task prev = head;

Task curr = head.next;

while (curr != null) {

if (curr.taskId == id) {

prev.next = curr.next;

return true;

}

prev = curr;

curr = curr.next;

}

return false;

}

// Main method

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

TaskManagementSystem tms = new TaskManagementSystem();

int choice;

do {

System.out.println("\n--- Task Management Menu ---");

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

System.out.println("2. Search Task");

System.out.println("3. Display All Tasks");

System.out.println("4. Delete Task");

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

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

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

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

int id = sc.nextInt();

sc.nextLine();

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

String name = sc.nextLine();

System.out.print("Enter Status (Pending/Completed): ");

String status = sc.nextLine();

tms.addTask(id, name, status);

break;

case 2:

System.out.print("Enter Task ID to search: ");

int searchId = sc.nextInt();

Task found = tms.searchTask(searchId);

if (found != null) found.display();

else System.out.println("Task not found.");

break;

case 3:

tms.traverseTasks();

break;

case 4:

System.out.print("Enter Task ID to delete: ");

int delId = sc.nextInt();

boolean deleted = tms.deleteTask(delId);

System.out.println(deleted ? "Task deleted." : "Task not found.");

break;

case 5:

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

break;

default:

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

}

} while (choice != 5);

}

}

**Output:**

--- Task Management Menu ---

1. Add Task

2. Search Task

3. Display All Tasks

4. Delete Task

5. Exit

Enter your choice: 1

Enter Task ID: 101

Enter Task Name: Designing a page

Enter Status (Pending/Completed): Pending

Task added.

--- Task Management Menu ---

1. Add Task

2. Search Task

3. Display All Tasks

4. Delete Task

5. Exit

Enter your choice: 1

Enter Task ID: 102

Enter Task Name: Embed into a softaware

Enter Status (Pending/Completed): Completed

Task added.

--- Task Management Menu ---

1. Add Task

2. Search Task

3. Display All Tasks

4. Delete Task

5. Exit

Enter your choice: 2

Enter Task ID to search: 102

TaskID: 102, Name: Embed into a softaware, Status: Completed

--- Task Management Menu ---

1. Add Task

2. Search Task

3. Display All Tasks

4. Delete Task

5. Exit

Enter your choice: 3

TaskID: 101, Name: Designing a page, Status: Pending

TaskID: 102, Name: Embed into a softaware, Status: Completed

--- Task Management Menu ---

1. Add Task

2. Search Task

3. Display All Tasks

4. Delete Task

5. Exit

Enter your choice: 4

Enter Task ID to delete: 102

Task deleted.

--- Task Management Menu ---

1. Add Task

2. Search Task

3. Display All Tasks

4. Delete Task

5. Exit

Enter your choice: 5

Exiting...

**Exercise 6: Library Management System**

**Implementation:**

public class Book {

int bookId;

String title;

String author;

public Book(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

public void display() {

System.out.println("Book ID: " + bookId + ", Title: " + title + ", Author: " + author);

}

}

//LibraryManagementSystem class

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class LibraryManagementSystem {

static Book[] books = {

new Book(101, "Data Structures", "Mark Allen"),

new Book(102, "The Alchemist", "Paulo Coelho"),

new Book(103, "C Programming", "Dennis Ritchie"),

new Book(104, "The Hobbit", "J.R.R. Tolkien"),

new Book(105, "Algorithms", "Robert Sedgewick")

};

// Linear search by title

public static Book linearSearchByTitle(String title) {

for (Book book : books) {

if (book.title.equalsIgnoreCase(title)) {

return book;

}

}

return null;

}

// Binary search by title (requires sorted array)

public static Book binarySearchByTitle(String title) {

Book[] sortedBooks = books.clone();

Arrays.sort(sortedBooks, Comparator.comparing(b -> b.title.toLowerCase()));

int left = 0;

int right = sortedBooks.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int compare = title.compareToIgnoreCase(sortedBooks[mid].title);

if (compare == 0) {

return sortedBooks[mid];

} else if (compare < 0) {

right = mid - 1;

} else {

left = mid + 1;

}

}

return null;

}

// Display all books

public static void displayBooks() {

for (Book book : books) {

book.display();

}

}

// Main program

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Library Management Menu ---");

System.out.println("1. Display All Books");

System.out.println("2. Search by Title (Linear Search)");

System.out.println("3. Search by Title (Binary Search - Sorted)");

System.out.println("4. Exit");

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

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

displayBooks();

break;

case 2:

System.out.print("Enter title to search (linear): ");

String title1 = sc.nextLine();

Book result1 = linearSearchByTitle(title1);

if (result1 != null)

result1.display();

else

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

break;

case 3:

System.out.print("Enter title to search (binary): ");

String title2 = sc.nextLine();

Book result2 = binarySearchByTitle(title2);

if (result2 != null)

result2.display();

else

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

break;

case 4:

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

break;

default:

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

}

} while (choice != 4);

}

}

**Output:**

--- Library Management Menu ---

1. Display All Books

2. Search by Title (Linear Search)

3. Search by Title (Binary Search - Sorted)

4. Exit

Enter your choice: 1

Book ID: 101, Title: Data Structures, Author: Mark Allen

Book ID: 102, Title: The Alchemist, Author: Paulo Coelho

Book ID: 103, Title: C Programming, Author: Dennis Ritchie

Book ID: 104, Title: The Hobbit, Author: J.R.R. Tolkien

Book ID: 105, Title: Algorithms, Author: Robert Sedgewick

--- Library Management Menu ---

1. Display All Books

2. Search by Title (Linear Search)

3. Search by Title (Binary Search - Sorted)

4. Exit

Enter your choice: 2

Enter title to search (linear): 3

Book not found.

--- Library Management Menu ---

1. Display All Books

2. Search by Title (Linear Search)

3. Search by Title (Binary Search - Sorted)

4. Exit

Enter your choice: 4

Exiting...

PS C:\Users\suman\OneDrive\Desktop\book>

**Exercise 7: Financial Forecasting**

**Implementation:**

import java.util.Scanner;

public class FinancialForecasting {

public static double futureValue(double principal, double rate, int years) {

double result = principal;

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

result \*= (1 + rate);

}

return result;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter current amount (principal): ");

double principal = sc.nextDouble();

System.out.print("Enter annual growth rate (in %): ");

double ratePercent = sc.nextDouble();

System.out.print("Enter number of years: ");

int years = sc.nextInt();

double rate = ratePercent / 100.0; // Convert to decimal

double result = futureValue(principal, rate, years);

System.out.printf("Predicted future value after %d years: %.2f\n", years, result);

}

}

**Output:**

Enter current amount (principal): 100000

Enter annual growth rate (in %): 10

Enter number of years: 3

Predicted future value after 3 years: 133100.00