**Exercise 1: Inventory Management System**

**InventoryManager.java**

package com.inventory;

import java.util.HashMap;

public class InventoryManager {

private HashMap<String, Product> inventory;

public InventoryManager() {

inventory = new HashMap<>();

}

// Add product

public void addProduct(Product product) {

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

System.***out***.println("Product already exists.");

} else {

inventory.put(product.getProductId(), product);

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

}

}

// Update product

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

Product product = inventory.get(productId);

if (product != null) {

product.setQuantity(quantity);

product.setPrice(price);

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

} else {

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

}

}

// Delete product

public void deleteProduct(String productId) {

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

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

} else {

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

}

}

// Display all

public void displayInventory() {

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

System.***out***.println(p);

}

}

}

**Main.java**

package com.inventory;

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

Product p1 = new Product("P101", "Keyboard", 20, 1500);

Product p2 = new Product("P102", "Mouse", 50, 700);

manager.addProduct(p1);

manager.addProduct(p2);

manager.displayInventory();

manager.updateProduct("P101", 15, 1450);

manager.deleteProduct("P102");

manager.displayInventory();

}

}

**Product.java**

package com.inventory;

public class Product {

private String productId;

private String productName;

private int quantity;

private double price;

// Constructor

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

// Getters and Setters

public String getProductId() { return productId; }

public String getProductName() { return productName; }

public int getQuantity() { return quantity; }

public double getPrice() { return price; }

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

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

*@Override*

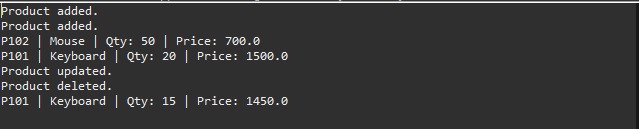
public String toString() {

return productId + " | " + productName + " | Qty: " + quantity + " | Price: " + price;

}

}

**OUTPUT**

****

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

**BinarySearch.java**

**package com.ecommerce;**

**import java.util.Arrays;**

**import java.util.Comparator;**

**public class BinarySearch {**

**public static Product binarySearch(Product[] products, String targetName) {**

**// Binary Search requires sorted data**

**Arrays.sort(products, Comparator.comparing(Product::getProductName, String.CASE\_INSENSITIVE\_ORDER));**

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

**while (left <= right) {**

**int mid = (left + right) / 2;**

**int compare = targetName.compareToIgnoreCase(products[mid].getProductName());**

**if (compare == 0) return products[mid];**

**else if (compare < 0) right = mid - 1;**

**else left = mid + 1;**

**}**

**return null;**

**}**

**}**

**LinearSearch.java**

package com.ecommerce;

public class LinearSearch {

public static Product linearSearch(Product[] products, String targetName) {

for (Product p : products) {

if (p.getProductName().equalsIgnoreCase(targetName)) {

return p;

}

}

return null;

}

}

**Main.java**

package com.ecommerce;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product("P001", "Shoes", "Footwear"),

new Product("P002", "Watch", "Accessories"),

new Product("P003", "T-shirt", "Clothing"),

new Product("P004", "Laptop", "Electronics"),

new Product("P005", "Bag", "Travel")

};

// Linear Search

Product result1 = LinearSearch.*linearSearch*(products, "Laptop");

System.***out***.println("Linear Search: " + (result1 != null ? result1 : "Product not found"));

// Binary Search

Product result2 = BinarySearch.*binarySearch*(products, "Laptop");

System.***out***.println("Binary Search: " + (result2 != null ? result2 : "Product not found"));

}

}

**Product.java**

package com.ecommerce;

public class Product {

String productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String getProductName() {

return productName;

}

*@Override*

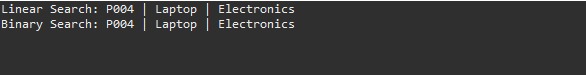
public String toString() {

return productId + " | " + productName + " | " + category;

}

}

**OUTPUT**

****

**Exercise 3: Sorting Customer Orders**

**Order.java**

package com.sort;

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 + ", Customer: " + customerName + ", Total: $" + totalPrice);

}

}

**OrderSorter.java**

package com.sort;

public class OrderSorter {

// Bubble Sort

public static void bubbleSort(Order[] orders) {

int n = orders.length;

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

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

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

Order temp = orders[j];

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

orders[j + 1] = temp;

}

}

}

}

// Quick Sort

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

if (low < high) {

int pivotIndex = *partition*(orders, low, high);

*quickSort*(orders, low, pivotIndex - 1);

*quickSort*(orders, pivotIndex + 1, high);

}

}

private static int partition(Order[] orders, int low, int high) {

double pivot = orders[high].totalPrice;

int i = low - 1;

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

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

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

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

orders[high] = temp;

return i + 1;

}

// Main method

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Alice", 250.0),

new Order(102, "Bob", 120.0),

new Order(103, "Charlie", 340.5),

new Order(104, "Diana", 99.9),

new Order(105, "Ethan", 180.0)

};

System.***out***.println("Original Orders:");

for (Order o : orders) o.display();

// Bubble Sort

*bubbleSort*(orders);

System.***out***.println("\nSorted by Bubble Sort:");

for (Order o : orders) o.display();

// Reset and apply Quick Sort

orders = new Order[]{

new Order(101, "Alice", 250.0),

new Order(102, "Bob", 120.0),

new Order(103, "Charlie", 340.5),

new Order(104, "Diana", 99.9),

new Order(105, "Ethan", 180.0)

};

*quickSort*(orders, 0, orders.length - 1);

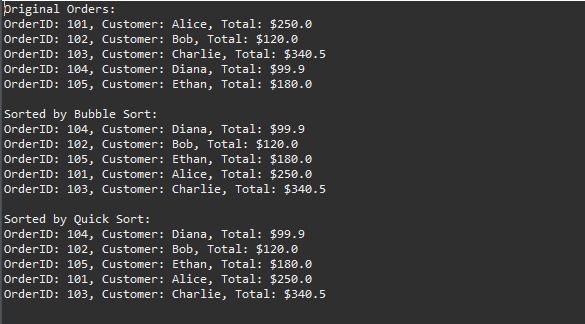
System.***out***.println("\nSorted by Quick Sort:");

for (Order o : orders) o.display();

}

}

**OUTPUT**

****

**Exercise 4: Employee Management System**

**Employee.java**

package com.employee;

public 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);

}

}

**EmployeeManager.java**

package com.employee;

import java.util.Scanner;

public class EmployeeManager {

static Employee[] *employees* = new Employee[100];

static int *count* = 0;

// Add employee

public static void addEmployee(Employee emp) {

if (*count* < *employees*.length) {

*employees*[*count*++] = emp;

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

} else {

System.***out***.println("Array is full.");

}

}

// Search employee by ID

public static void searchEmployee(int id) {

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

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

*employees*[i].display();

return;

}

}

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

}

// Traverse (display all)

public static void traverseEmployees() {

if (*count* == 0) {

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

return;

}

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

*employees*[i].display();

}

}

// Delete employee by ID

public static void 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;

System.***out***.println("Employee deleted.");

return;

}

}

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

}

// Main method

public static void main(String[] args) {

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

while (true) {

System.***out***.println("\n1. Add Employee\n2. Search Employee\n3. Display All Employees\n4. Delete Employee\n5. Exit");

System.***out***.print("Choose an option: ");

int option = sc.nextInt();

switch (option) {

case 1:

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

int id = sc.nextInt();

sc.nextLine(); // consume newline

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();

*addEmployee*(new Employee(id, name, position, salary));

break;

case 2:

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

*searchEmployee*(sc.nextInt());

break;

case 3:

*traverseEmployees*();

break;

case 4:

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

*deleteEmployee*(sc.nextInt());

break;

case 5:

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

return;

default:

System.***out***.println("Invalid option.");

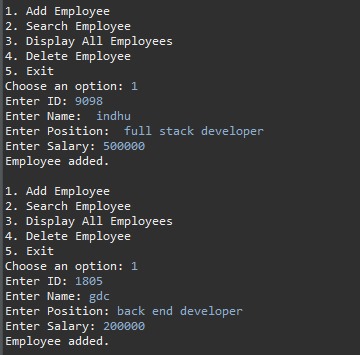
}

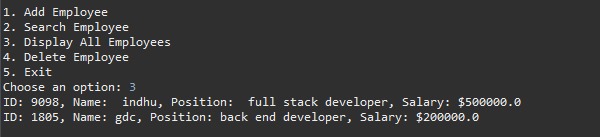
}

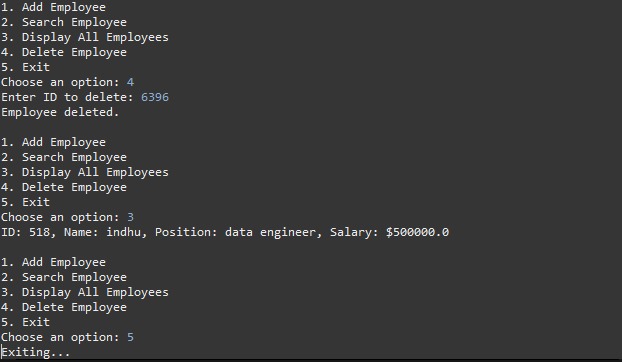
}

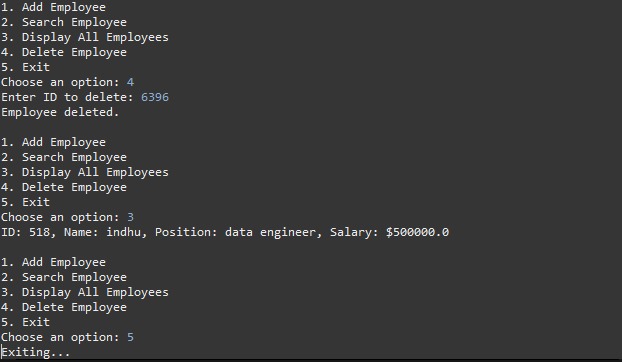
}

**OUTPUT**

****

****

****

****

**Exercise 5: Task Management System**

**Node.java**

package com.task;

class Node {

Task task;

Node next;

public Node(Task task) {

this.task = task;

this.next = null;

}

}

**Task.java**

package com.task;

public class Task {

int taskId;

String taskName;

String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public void display() {

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

}

}

**TaskManager.java**

package com.task;

import java.util.Scanner;

public class TaskManager {

static Node *head* = null;

// Add task at end

public static void addTask(Task task) {

Node newNode = new Node(task);

if (*head* == null) {

*head* = newNode;

} else {

Node temp = *head*;

while (temp.next != null) temp = temp.next;

temp.next = newNode;

}

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

}

// Search task by ID

public static void searchTask(int id) {

Node temp = *head*;

while (temp != null) {

if (temp.task.taskId == id) {

temp.task.display();

return;

}

temp = temp.next;

}

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

}

// Traverse tasks

public static void traverseTasks() {

Node temp = *head*;

if (temp == null) {

System.***out***.println("No tasks to display.");

return;

}

while (temp != null) {

temp.task.display();

temp = temp.next;

}

}

// Delete task by ID

public static void deleteTask(int id) {

if (*head* == null) {

System.***out***.println("Task list is empty.");

return;

}

if (*head*.task.taskId == id) {

*head* = *head*.next;

System.***out***.println("Task deleted.");

return;

}

Node temp = *head*;

while (temp.next != null && temp.next.task.taskId != id) {

temp = temp.next;

}

if (temp.next == null) {

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

} else {

temp.next = temp.next.next;

System.***out***.println("Task deleted.");

}

}

// Main method

public static void main(String[] args) {

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

while (true) {

System.***out***.println("\n1. Add Task\n2. Search Task\n3. View All Tasks\n4. Delete Task\n5. Exit");

System.***out***.print("Choose an option: ");

int option = sc.nextInt();

switch (option) {

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: ");

String status = sc.nextLine();

*addTask*(new Task(id, name, status));

break;

case 2:

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

*searchTask*(sc.nextInt());

break;

case 3:

*traverseTasks*();

break;

case 4:

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

*deleteTask*(sc.nextInt());

break;

case 5:

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

return;

default:

System.***out***.println("Invalid option.");

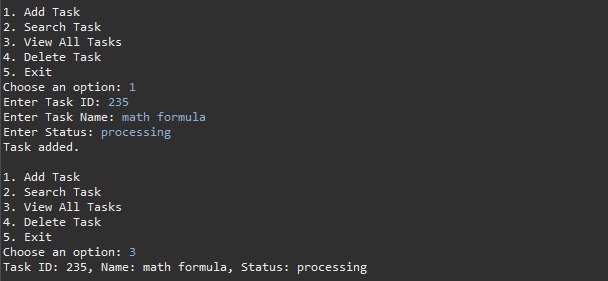
}

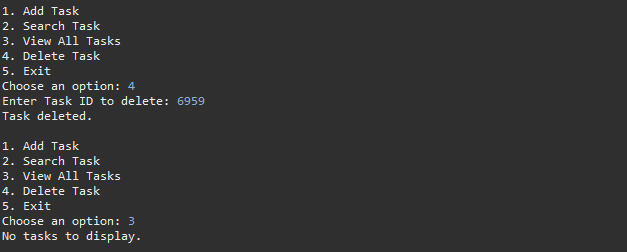
}

}

}

**OUTPUT**

****

****

**Exercise 6: Library Management System**

**LibrabryManagementSystem.java**

package com.library;

import java.util.\*;

// Book class with attributes

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);

}

}

public class LibraryManagementSystem {

// Linear Search Method

public static Book linearSearch(Book[] books, String targetTitle) {

for (Book book : books) {

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

return book;

}

}

return null;

}

// Binary Search Method (assumes sorted by title)

public static Book binarySearch(Book[] books, String targetTitle) {

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

while (left <= right) {

int mid = (left + right) / 2;

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

if (compare == 0) return books[mid];

else if (compare < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

// Sample book data

Book[] books = {

new Book(101, "Java Basics", "Indhumathi"),

new Book(102, "Python Pro", "Rahul"),

new Book(103, "Data Structures", "Priya"),

new Book(104, "C Programming", "Ravi"),

new Book(105, "Algorithms", "Deepa")

};

// Sorting array for binary search

Arrays.*sort*(books, Comparator.*comparing*(book -> book.title.toLowerCase()));

// User Input

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

String titleToSearch = scanner.nextLine();

// Linear Search

Book foundLinear = *linearSearch*(books, titleToSearch);

if (foundLinear != null) {

System.***out***.println("\n Book found using Linear Search:");

foundLinear.display();

} else {

System.***out***.println("\n Book not found using Linear Search.");

}

// Binary Search

Book foundBinary = *binarySearch*(books, titleToSearch);

if (foundBinary != null) {

System.***out***.println("\n Book found using Binary Search:");

foundBinary.display();

} else {

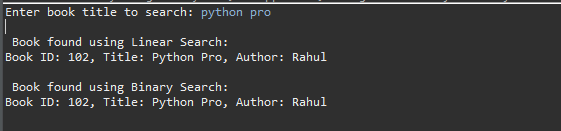
System.***out***.println("\n Book not found using Binary Search.");

}

}

}

**OUTPUT**

****

**Exercise 7: Financial Forecasting**

**FinancialForecast.java**

package com.forecast;

import java.util.Scanner;

public class FinancialForecast {

// Recursive method to calculate future value

public static double forecastRecursive(int year, double initialValue, double growthRate) {

// Base case: 0 years ahead returns current value

if (year == 0) {

return initialValue;

}

// Recursive step

return *forecastRecursive*(year - 1, initialValue, growthRate) \* (1 + growthRate);

}

// Optimized Iterative method to calculate future value

public static double forecastIterative(int year, double initialValue, double growthRate) {

double value = initialValue;

for (int i = 1; i <= year; i++) {

value \*= (1 + growthRate);

}

return value;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

// User input

System.***out***.print("Enter initial value (e.g., investment or revenue): ₹");

double initialValue = scanner.nextDouble();

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

double growthRatePercent = scanner.nextDouble();

double growthRate = growthRatePercent / 100; // Convert to decimal

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

int years = scanner.nextInt();

// Recursive forecast

double recursiveResult = *forecastRecursive*(years, initialValue, growthRate);

System.***out***.printf("\nFuture Value (Recursive) after %d years = ₹%.2f", years, recursiveResult);

// Iterative forecast

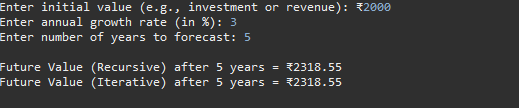
double iterativeResult = *forecastIterative*(years, initialValue, growthRate);

System.***out***.printf("\nFuture Value (Iterative) after %d years = ₹%.2f\n", years, iterativeResult);

}

}

**OUTPUT**

****