**Exercise 1: Inventory Management System**

**Scenario:**

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

**Program: 🡪 Product.java**

package product;

public class Product {

private String productId;

private String productName;

private int quantity;

private double price;

public Product(String id, String name, int qty, double price) {

this.productId = id;

this.productName = name;

this.quantity = qty;

this.price = price;

}

public String getProductId() { return productId; }

public void setProductId(String id) { this.productId = id; }

public String getProductName() { return productName; }

public void setProductName(String name) { this.productName = name; }

public int getQuantity() { return quantity; }

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

public double getPrice() { return price; }

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

@Override

public String toString() {

return "ID: " + productId + " | " + productName +

" | Qty: " + quantity + " | Price: $" + price;

}

}

**Program: 🡪 Main.java**

package product;

public class Main {

public static void main(String[] args) {

Product p = new Product("P101", "Laptop", 5, 59999.99);

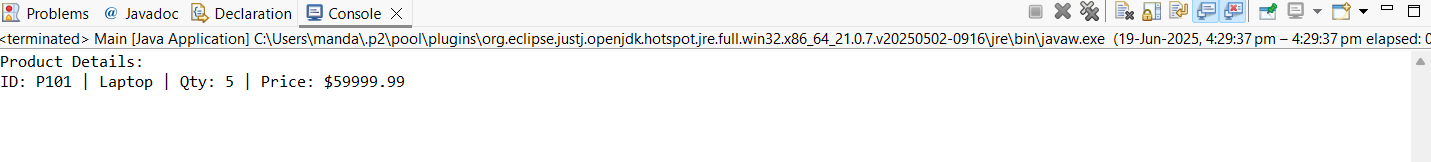
System.*out*.println("Product Details:");

System.*out*.println(p);

}

}

**OUTPUT:**



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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Program: 🡪Product.java**

package searchengine;

public class Product {

private int productId;

private String productName;

private String category;

public Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

public int getProductId() { return productId; }

public String getProductName() { return productName; }

public String getCategory() { return category; }

@Override

public String toString() {

return "[" + productId + "] " + productName + " (" + category + ")";

}

}

**Program: 🡪 main()**

package searchengine;

import java.util.Arrays;

import java.util.Comparator;

public class ProductSearch {

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shirt", "Clothing"),

new Product(103, "Mobile", "Electronics"),

new Product(104, "Shoes", "Footwear"),

new Product(105, "Watch", "Accessories")

};

System.out.println("Linear Search for 'Mobile':");

Product found1 = linearSearch(products, "Mobile");

System.out.println(found1 != null ? found1 : "Not found");

Arrays.sort(products, Comparator.comparing(Product::getProductName));

System.out.println("\nBinary Search for 'Mobile':");

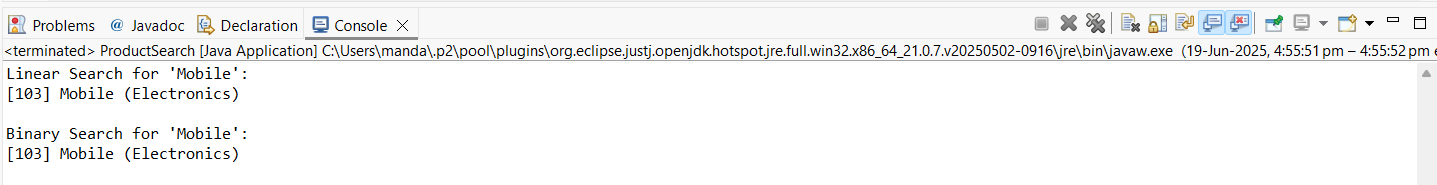
Product found2 = binarySearch(products, "Mobile");

System.out.println(found2 != null ? found2 : "Not found");

}

}

**OUTPUT:**



**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**Program: 🡪 CustomerOrder.java**

package ordersystem;

public class CustomerOrder {

private String orderId;

private String customerName;

private double totalPrice;

public CustomerOrder(String orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public String getOrderId() { return orderId; }

public String getCustomerName() { return customerName; }

public double getTotalPrice() { return totalPrice; }

@Override

public String toString() {

return "Order ID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice;

}

}

**Program: 🡪 main()**

package ordersystem;

import java.util.Arrays;

import java.util.Comparator;

public class OrderSorter {

public static void main(String[] args) {

// Step 1: Create an array of CustomerOrder

CustomerOrder[] orders = {

new CustomerOrder("O1001", "Alice", 4500.75),

new CustomerOrder("O1002", "Bob", 999.99),

new CustomerOrder("O1003", "Charlie", 12000.50),

new CustomerOrder("O1004", "David", 7000.00)

};

// Step 2: Sort orders by totalPrice in descending order (high-value first)

Arrays.sort(orders, new Comparator<CustomerOrder>() {

public int compare(CustomerOrder o1, CustomerOrder o2) {

return Double.compare(o2.getTotalPrice(), o1.getTotalPrice());

}

});

// Step 3: Print sorted orders

System.out.println("Sorted Customer Orders (High to Low):");

for (CustomerOrder order : orders) {

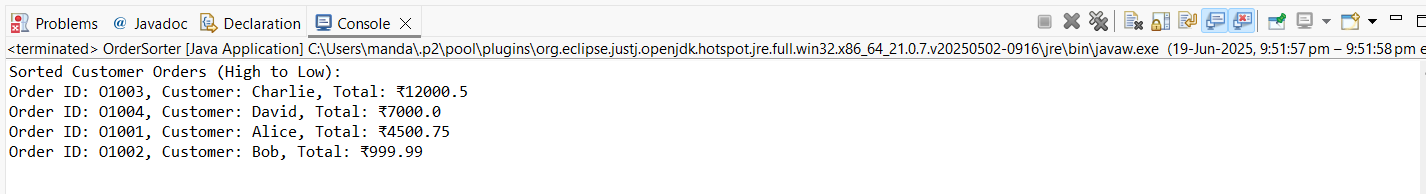
System.out.println(order);

}

}

}

**OUTPUT:**



**Exercise 4: Employee Management System**

**Scenario:**

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**PROGRAM: 🡪 Employee.java**

package employee;

public class Employee {

private int empId;

private String name;

private String department;

private double salary;

public Employee(int empId, String name, String department, double salary) {

this.empId = empId;

this.name = name;

this.department = department;

this.salary = salary;

}

public int getEmpId() { return empId; }

public String getName() { return name; }

public String getDepartment() { return department; }

public double getSalary() { return salary; }

@Override

public String toString() {

return "ID: " + empId + ", Name: " + name + ", Dept: " + department + ", Salary: ₹" + salary;

}

}

**Program: 🡪 main()**

package employee;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class EmployeeManager {

public static void main(String[] args) {

// Step 1: Create a list of employees

ArrayList<Employee> employees = new ArrayList<>();

employees.add(new Employee(101, "Alice", "HR", 45000));

employees.add(new Employee(102, "Bob", "IT", 60000));

employees.add(new Employee(103, "Charlie", "Finance", 55000));

employees.add(new Employee(104, "David", "IT", 75000));

employees.add(new Employee(105, "Eve", "Sales", 40000));

// Step 2: Display all employees

System.out.println("Employee Records:");

for (Employee emp : employees) {

System.out.println(emp);

}

// Step 3: Sort employees by salary (high to low)

Collections.sort(employees, new Comparator<Employee>() {

public int compare(Employee e1, Employee e2) {

return Double.compare(e2.getSalary(), e1.getSalary());

}

});

// Step 4: Display sorted employees

System.out.println("\nEmployees sorted by salary (high to low):");

for (Employee emp : employees) {

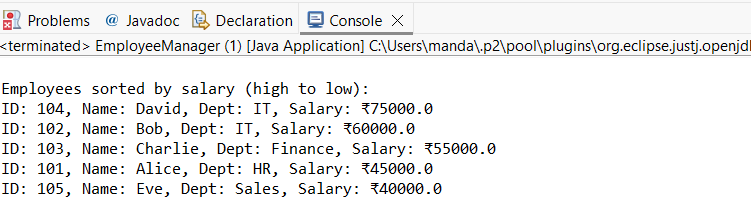
System.out.println(emp);

}

}

}

**OUTPUT:**



**Exercise 5: Task Management System**

**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**PROGRAM: 🡪 Task.java**

package taskmanager;

public class Task {

private String title;

private String description;

public Task(String title, String description) {

this.title = title;

this.description = description;

}

public String getTitle() { return title; }

public String getDescription() { return description; }

@Override

public String toString() {

return "Task: " + title + " - " + description;

}

}

**Program: 🡪 main()**

package taskmanager;

import java.util.LinkedList;

import java.util.Scanner;

public class TaskManager {

public static void main(String[] args) {

LinkedList<Task> taskList = new LinkedList<>();

Scanner scanner = new Scanner(System.in);

while (true) {

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

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

System.out.println("2. Delete Task by Title");

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

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

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

int choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

System.out.print("Enter task title: ");

String title = scanner.nextLine();

System.out.print("Enter task description: ");

String desc = scanner.nextLine();

taskList.add(new Task(title, desc));

System.out.println("✅ Task added.");

break;

case 2:

System.out.print("Enter title of task to delete: ");

String delTitle = scanner.nextLine();

boolean removed = taskList.removeIf(task -> task.getTitle().equalsIgnoreCase(delTitle));

if (removed)

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

else

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

break;

case 3:

System.out.println("📋 Task List:");

if (taskList.isEmpty()) {

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

} else {

for (Task t : taskList) {

System.out.println(t);

}

}

break;

case 4:

System.out.println("👋 Exiting Task Manager.");

scanner.close();

System.exit(0);

default:

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

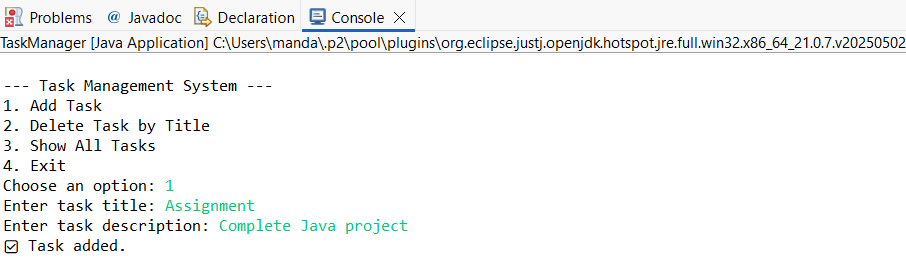
}

}

}

}

**OUTPUT:**



**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system where users can search for books by title or author.

**PROGRAM: 🡪 Book.java**

package library;

public class Book {

private String title;

private String author;

private String isbn;

public Book(String title, String author, String isbn) {

this.title = title;

this.author = author;

this.isbn = isbn;

}

public String getTitle() { return title; }

public String getAuthor() { return author; }

public String getIsbn() { return isbn; }

@Override

public String toString() {

return "📘 Title: " + title + " | Author: " + author + " | ISBN: " + isbn;

}

}

**PROGRAM: 🡪 MAIN()**

**LibraryManager.java**

package library;

import java.util.ArrayList;

import java.util.Scanner;

public class LibraryManager {

public static void main(String[] args) {

ArrayList<Book> books = new ArrayList<>();

Scanner scanner = new Scanner(System.in);

while (true) {

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

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

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

System.out.println("3. Search by Author");

System.out.println("4. Show All Books");

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

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

int choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

System.out.print("Enter Book Title: ");

String title = scanner.nextLine();

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

String author = scanner.nextLine();

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

String isbn = scanner.nextLine();

books.add(new Book(title, author, isbn));

System.out.println("✅ Book added successfully.");

break;

case 2:

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

String searchTitle = scanner.nextLine();

boolean foundTitle = false;

for (Book b : books) {

if (b.getTitle().equalsIgnoreCase(searchTitle)) {

System.out.println(b);

foundTitle = true;

}

}

if (!foundTitle) System.out.println("❌ No book found with that title.");

break;

case 3:

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

String searchAuthor = scanner.nextLine();

boolean foundAuthor = false;

for (Book b : books) {

if (b.getAuthor().equalsIgnoreCase(searchAuthor)) {

System.out.println(b);

foundAuthor = true;

}

}

if (!foundAuthor) System.out.println("❌ No book found by that author.");

break;

case 4:

System.out.println("📚 All Available Books:");

if (books.isEmpty()) {

System.out.println("No books in the library.");

} else {

for (Book b : books) {

System.out.println(b);

}

}

break;

case 5:

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

scanner.close();

System.exit(0);

default:

System.out.println("❌ Invalid option. Try again.");

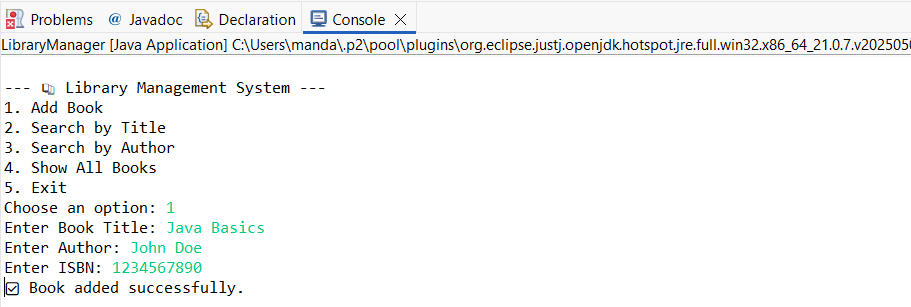
}

}

}

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**PROGRAM: 🡪 Forecaster.java**

package forecast;

import java.util.Scanner;

public class Forecaster {

public static double calculateSMA(double[] data, int period, int startIndex) {

if (startIndex + period > data.length) return 0;

double sum = 0;

for (int i = startIndex; i < startIndex + period; i++) {

sum += data[i];

}

return sum / period;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of past months of data: ");

int n = scanner.nextInt();

double[] revenue = new double[n];

System.out.println("Enter revenue for each month:");

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

System.out.print("Month " + (i + 1) + ": ₹");

revenue[i] = scanner.nextDouble();

}

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

int futureMonths = scanner.nextInt();

System.out.print("Enter moving average period (e.g., 3): ");

int period = scanner.nextInt();

System.out.println("\n📈 Forecasted Revenue:");

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

double forecast = calculateSMA(revenue, period, revenue.length - period);

System.out.printf("Month %d: ₹%.2f\n", n + i + 1, forecast);

double[] newRevenue = new double[revenue.length + 1];

System.arraycopy(revenue, 0, newRevenue, 0, revenue.length);

newRevenue[revenue.length] = forecast;

revenue = newRevenue;

}

scanner.close();

}

}

**OUTPUT:**

