**Superset ID: 6416838**

**Exercise 1: Inventory Management System**

**PROGRAM:**

import java.util.\*;

public class InventorySystem {

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

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

    public static void addProduct() {

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

        int id = sc.nextInt();

        sc.nextLine(); // clear buffer

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

        String name = sc.nextLine();

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

        int qty = sc.nextInt();

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

        double price = sc.nextDouble();

        if (!inventory.containsKey(id)) {

            inventory.put(id, new Product(id, name, qty, price));

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

        } else {

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

        }

    }

    public static void updateProduct() {

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

        int id = sc.nextInt();

        if (inventory.containsKey(id)) {

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

            int qty = sc.nextInt();

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

            double price = sc.nextDouble();

            Product p = inventory.get(id);

            p.quantity = qty;

            p.price = price;

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

        } else {

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

        }

    }

    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.");

        } else {

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

        }

    }

    public static void displayInventory() {

        if (inventory.isEmpty()) {

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

        } else {

            System.out.println("Inventory List:");

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

                System.out.println(p);

            }

        }

    }

    public static void main(String[] args) {

        while (true) {

            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. View Inventory");

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

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

            int 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..."); return;

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

            }

        }

    }

}

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;

    }

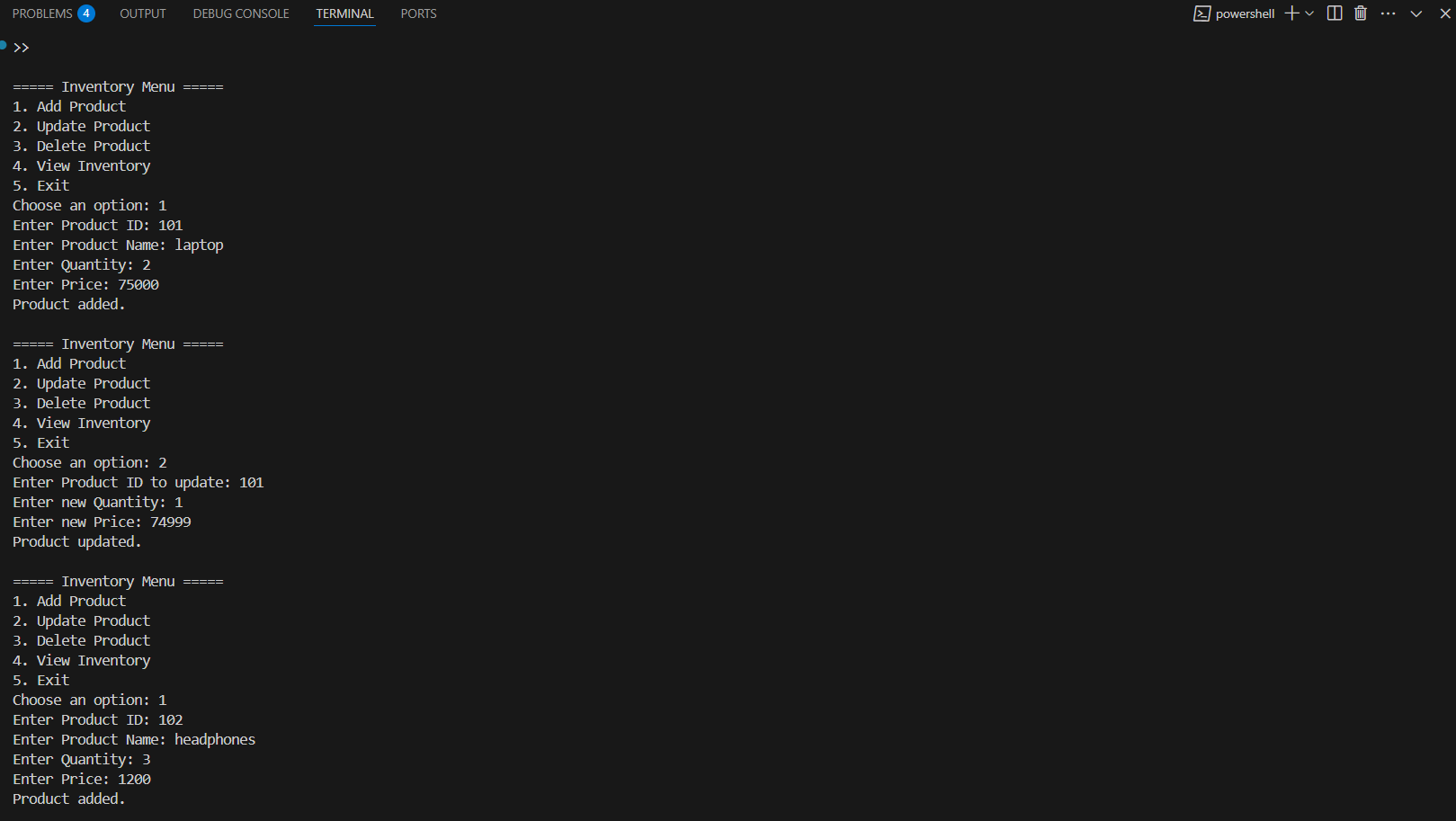
    public String toString() {

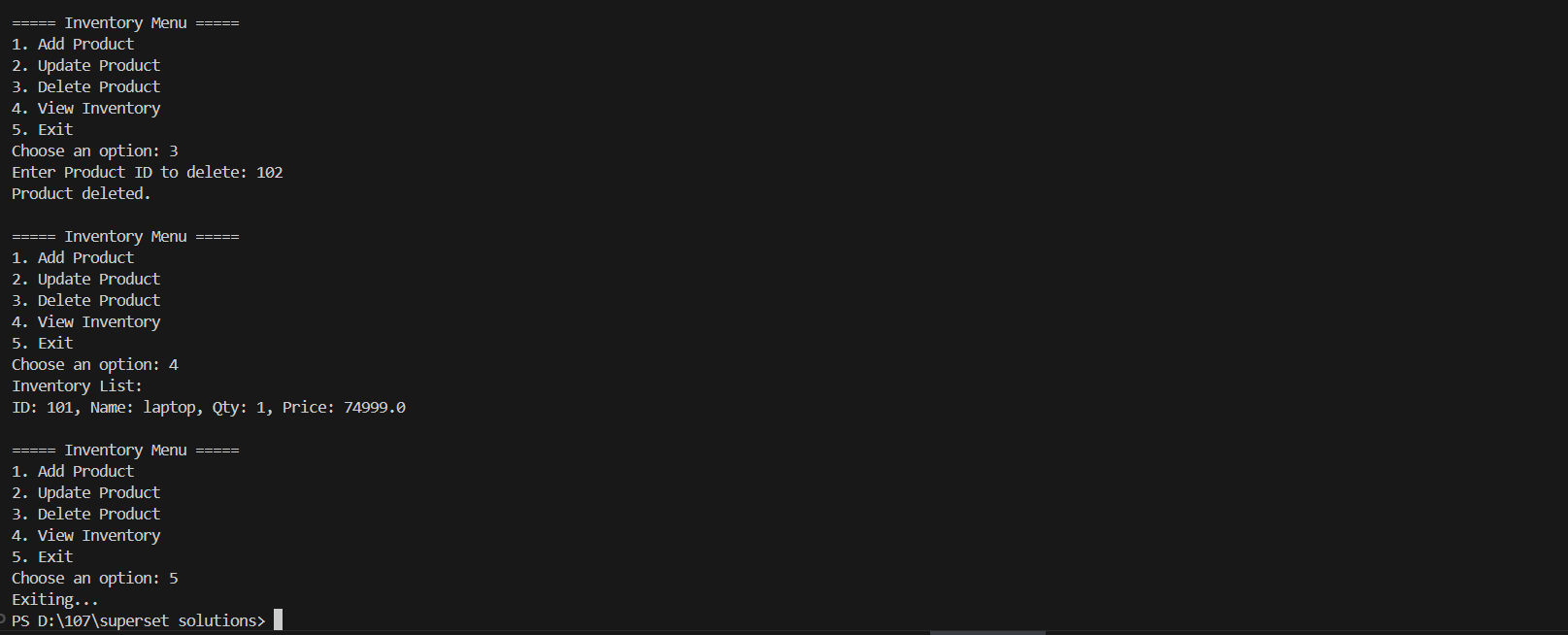
        return "ID: " + productId + ", Name: " + productName + ", Qty: " + quantity + ", Price: " + price;

    }

}

**OUTPUT:**

****

****

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

**PROGRAM:**

import java.util.\*;

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 String toString() {

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

public class ECommerceSearch {

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

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

while (left <= right) {

int mid = (left + right) / 2;

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

if (cmp == 0) {

return products[mid];

} else if (cmp < 0) {

right = mid - 1;

} else {

left = mid + 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Watch", "Accessories"),

new Product(4, "Smartphone", "Electronics"),

new Product(5, "T-shirt", "Fashion")

};

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name to search (Linear Search): ");

String search1 = sc.nextLine();

Product result1 = linearSearch(products, search1);

if (result1 != null) {

System.out.println("Found using Linear Search: " + result1);

} else {

System.out.println("Product not found using Linear Search.");

}

System.out.print("\nEnter product name to search (Binary Search): ");

String search2 = sc.nextLine();

Product result2 = binarySearch(products, search2);

if (result2 != null) {

System.out.println("Found using Binary Search: " + result2);

} else {

System.out.println("Product not found using Binary Search.");

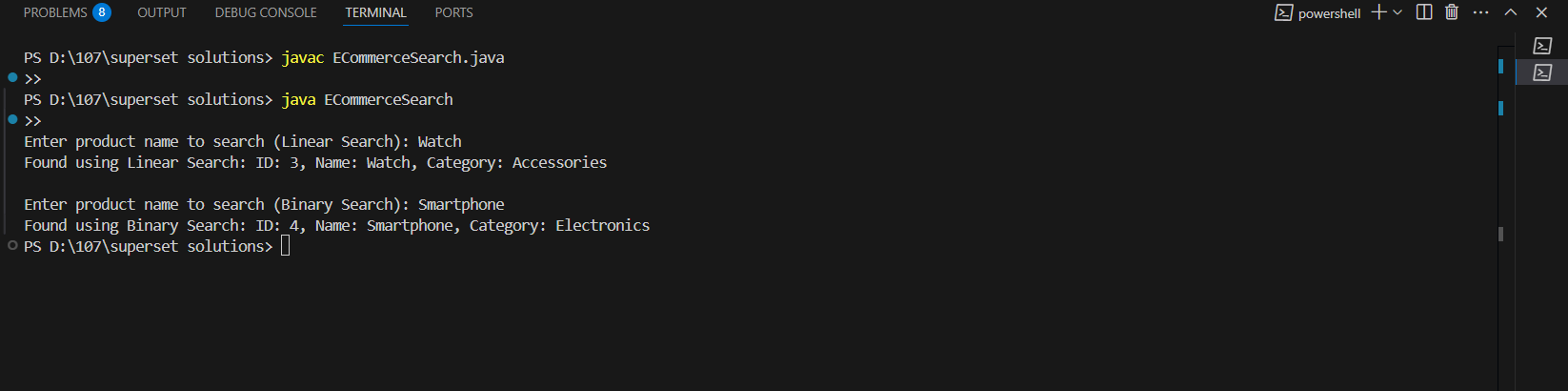
}

sc.close();

}

}

**OUTPUT:**

****

**Exercise 3: Sorting Customer Orders**

**PROGRAM:**

import java.util.\*;

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 String toString() {

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

}

}

public class OrderSorting {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

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

for (int j = 0; j < n - 1 - i; 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 quickSort(Order[] orders, int low, int high) {

if (low < high) {

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

quickSort(orders, low, pi - 1);

quickSort(orders, pi + 1, high);

}

}

public 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;

}

public static void displayOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

}

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Alice", 2500),

new Order(102, "Bob", 1200),

new Order(103, "Charlie", 4300),

new Order(104, "David", 3100)

};

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

displayOrders(orders);

// Bubble Sort

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

bubbleSort(orders);

displayOrders(orders);

// Reset original order

orders = new Order[]{

new Order(101, "Alice", 2500),

new Order(102, "Bob", 1200),

new Order(103, "Charlie", 4300),

new Order(104, "David", 3100)

};

// Quick Sort

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

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

displayOrders(orders);

}

}

**OUTPUT:**

****

**Exercise 4: Employee Management System**

**PROGRAM:**

import java.util.Scanner;

class Employee {

    int employeeId;

    String name;

    String position;

    double salary;

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

        this.employeeId = employeeId;

        this.name = name;

        this.position = position;

        this.salary = salary;

    }

    void display() {

        System.out.println("ID: " + employeeId + ", Name: " + name +

                           ", Position: " + position + ", Salary: " + salary);

    }

}

public class EmployeeManagement {

    static Employee[] employees = new Employee[100]; // max 100 employees

    static int count = 0;

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

    // Add employee

    static void addEmployee() {

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

        int id = sc.nextInt();

        sc.nextLine(); // clear buffer

        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();

        if (count < employees.length) {

            employees[count++] = new Employee(id, name, position, salary);

            System.out.println("Employee added successfully.\n");

        } else {

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

        }

    }

    // Search employee

    static void searchEmployee() {

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

        int id = sc.nextInt();

        boolean found = false;

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

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

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

                employees[i].display();

                found = true;

                break;

            }

        }

        if (!found) System.out.println("Employee not found.\n");

    }

    // Traverse employees

    static void traverseEmployees() {

        if (count == 0) {

            System.out.println("No employees to display.\n");

            return;

        }

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

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

            employees[i].display();

        }

        System.out.println();

    }

    // Delete employee

    static void deleteEmployee() {

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

        int id = sc.nextInt();

        boolean found = false;

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

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

                // Shift all elements to the left

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

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

                }

                employees[--count] = null;

                System.out.println("Employee deleted successfully.\n");

                found = true;

                break;

            }

        }

        if (!found) System.out.println("Employee not found.\n");

    }

    // Main menu

    public static void main(String[] args) {

        int choice;

        do {

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

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

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

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

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

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

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

            choice = sc.nextInt();

            System.out.println();

            switch (choice) {

                case 1:

                    addEmployee();

                    break;

                case 2:

                    searchEmployee();

                    break;

                case 3:

                    traverseEmployees();

                    break;

                case 4:

                    deleteEmployee();

                    break;

                case 5:

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

                    break;

                default:

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

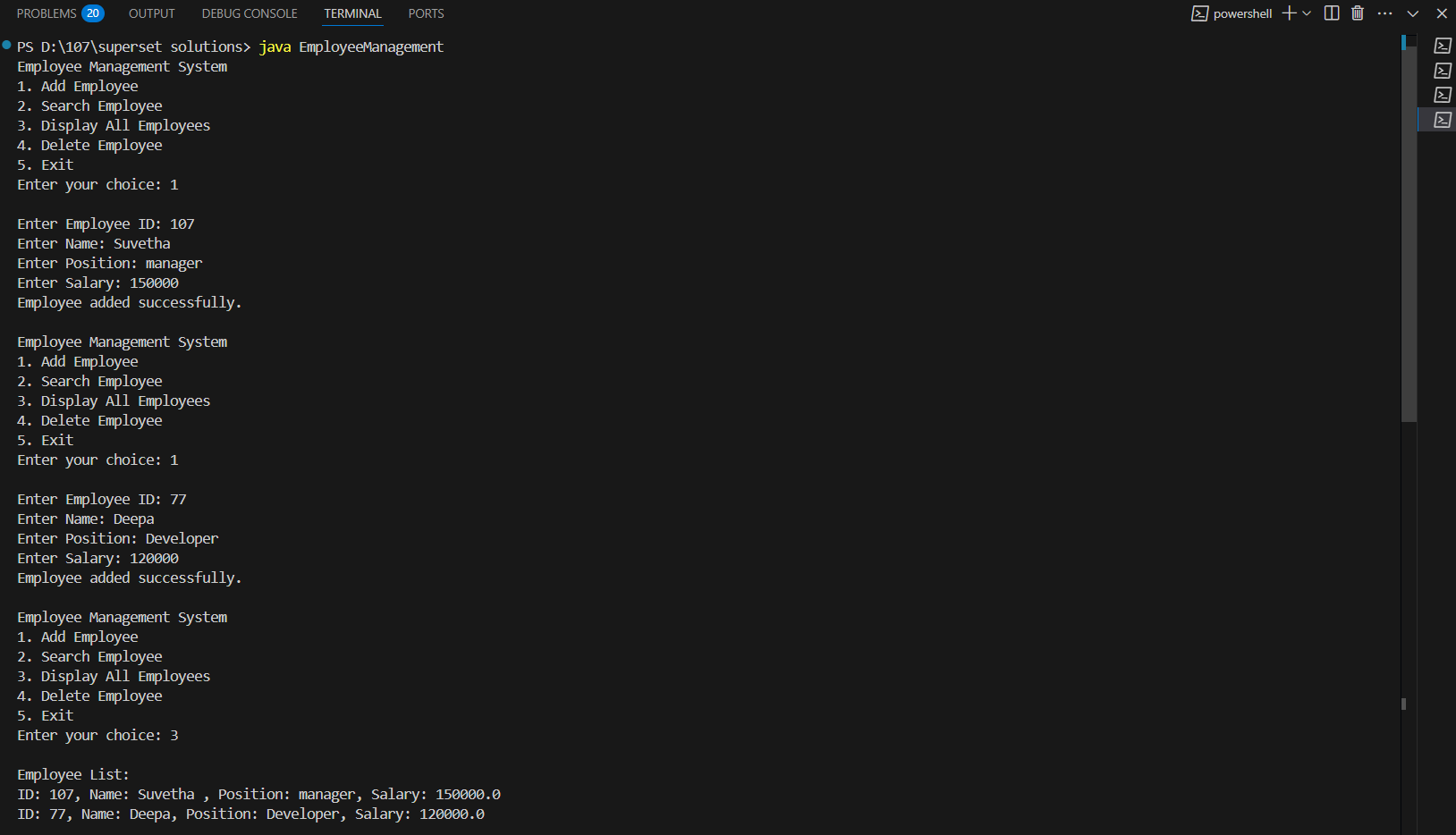
            }

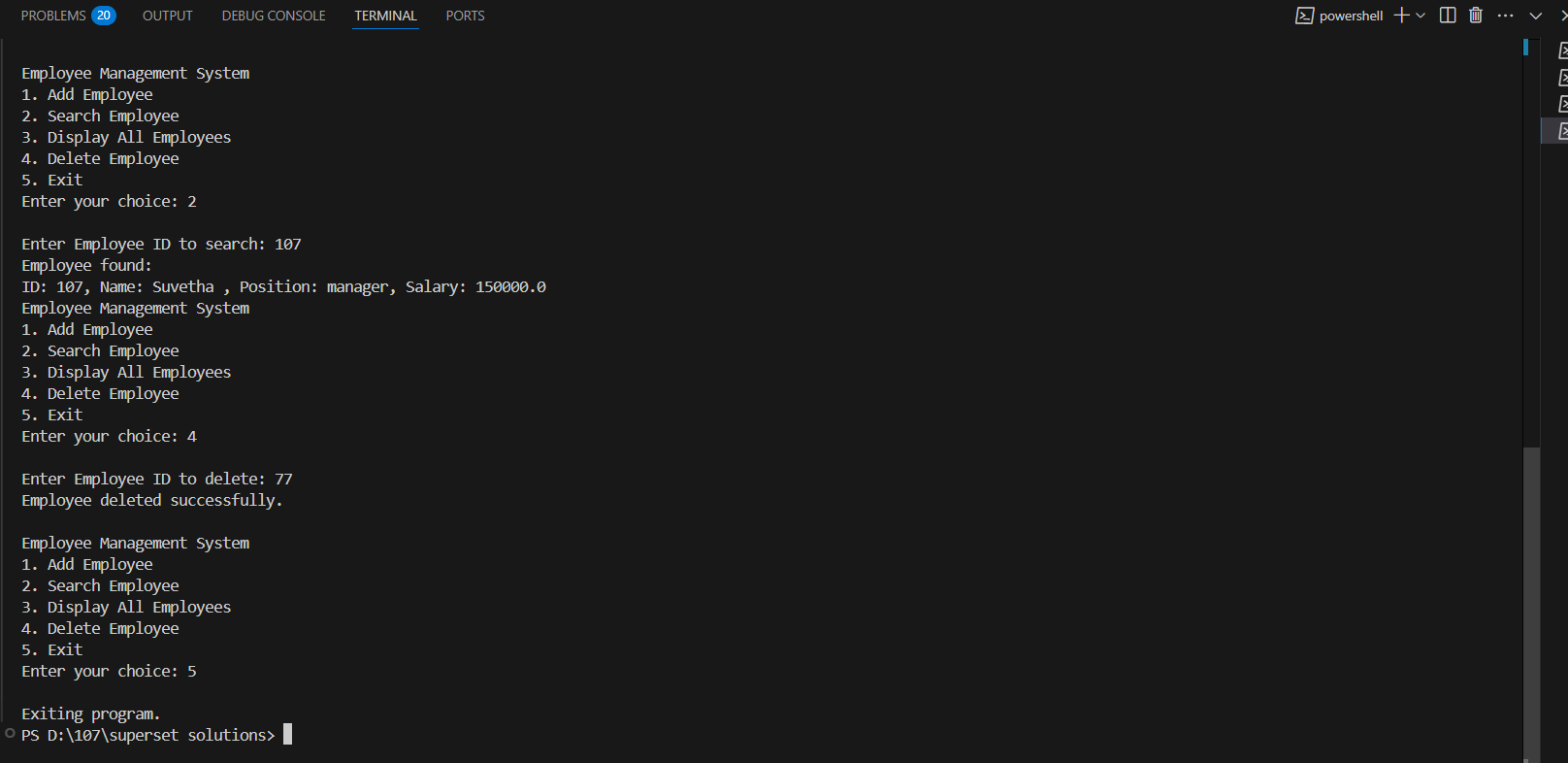
        } while (choice != 5);

    }

}

**OUTPUT:**

****



**Exercise 5: Task Management System**

**PROGRAM:**

import java.util.Scanner;

class Task {

    String taskName;

    String date;

    Task next;

    Task(String taskName, String date) {

        this.taskName = taskName;

        this.date = date;

        this.next = null;

    }

    void display() {

        System.out.println("Task: " + taskName + ", Date: " + date);

    }

}

public class TaskManagement {

    static Task head = null;

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

    // Add task at end

    static void addTask() {

        sc.nextLine(); // clear buffer

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

        String name = sc.nextLine();

        System.out.print("Enter Date (DD-MM-YYYY): ");

        String date = sc.nextLine();

        Task newTask = new Task(name, date);

        if (head == null) {

            head = newTask;

        } else {

            Task temp = head;

            while (temp.next != null) {

                temp = temp.next;

            }

            temp.next = newTask;

        }

        System.out.println("Task added successfully.\n");

    }

    // Search task by name

    static void searchTask() {

        sc.nextLine();

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

        String name = sc.nextLine();

        Task temp = head;

        while (temp != null) {

            if (temp.taskName.equalsIgnoreCase(name)) {

                System.out.println("Task found:");

                temp.display();

                return;

            }

            temp = temp.next;

        }

        System.out.println("Task not found.\n");

    }

    // Display all tasks

    static void displayTasks() {

        if (head == null) {

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

            return;

        }

        System.out.println("Task List:");

        Task temp = head;

        while (temp != null) {

            temp.display();

            temp = temp.next;

        }

        System.out.println();

    }

    // Delete task by name

    static void deleteTask() {

        sc.nextLine();

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

        String name = sc.nextLine();

        if (head == null) {

            System.out.println("No tasks to delete.\n");

            return;

        }

        if (head.taskName.equalsIgnoreCase(name)) {

            head = head.next;

            System.out.println("Task deleted successfully.\n");

            return;

        }

        Task prev = head;

        Task curr = head.next;

        while (curr != null) {

            if (curr.taskName.equalsIgnoreCase(name)) {

                prev.next = curr.next;

                System.out.println("Task deleted successfully.\n");

                return;

            }

            prev = curr;

            curr = curr.next;

        }

        System.out.println("Task not found.\n");

    }

    public static void main(String[] args) {

        int choice;

        do {

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

            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();

            System.out.println();

            switch (choice) {

                case 1: addTask(); break;

                case 2: searchTask(); break;

                case 3: displayTasks(); break;

                case 4: deleteTask(); break;

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

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

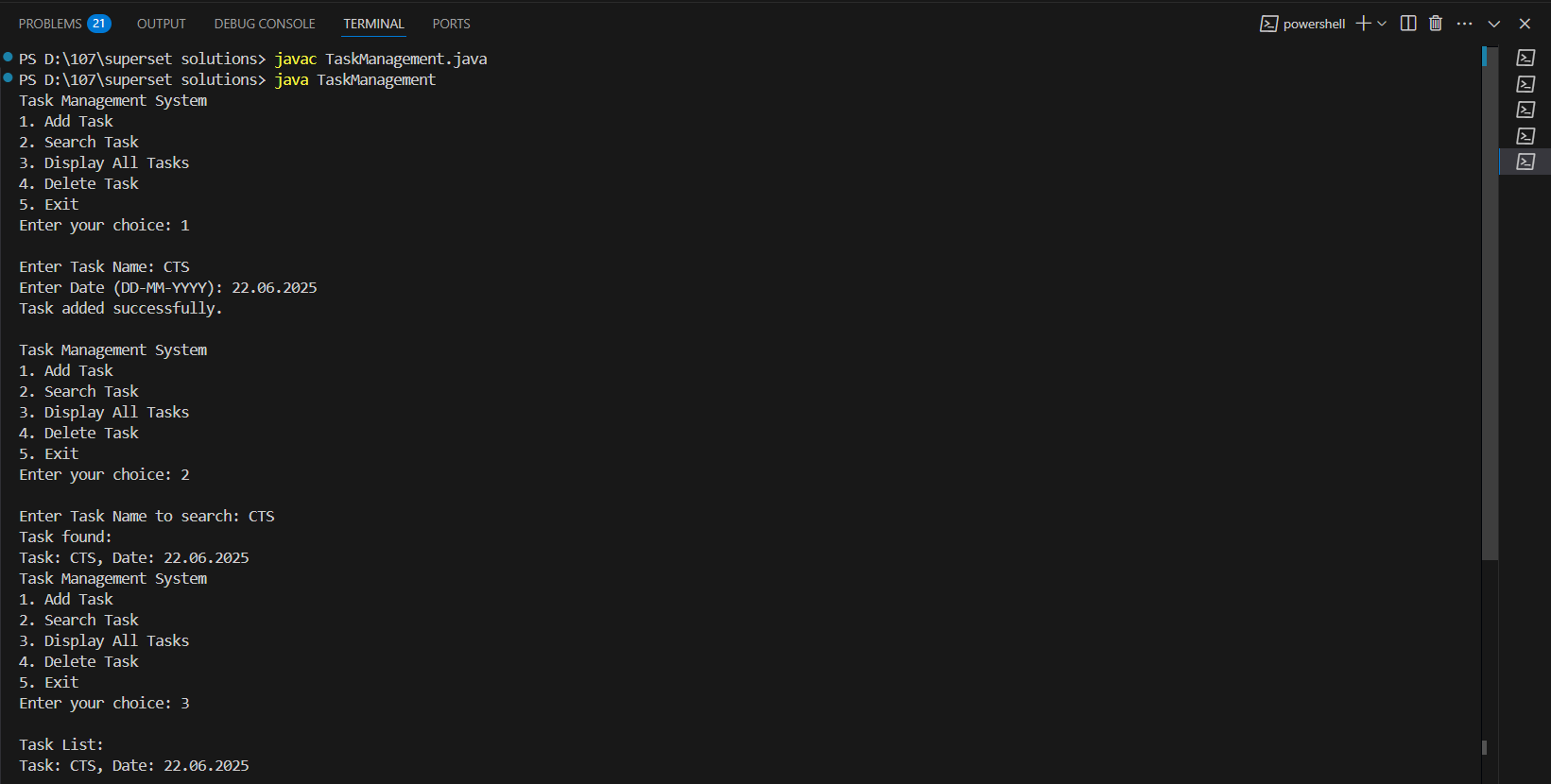
            }

        } while (choice != 5);

    }

}

**OUTPUT:**



**Exercise 6: Library Management System**

PROGRAM

import java.util.\*;

class Book {

    int bookId;

    String title;

    String author;

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

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    void display() {

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

    }

}

public class BookSearch {

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

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

    // Sample books (hardcoded)

    static void loadBooks() {

        books.add(new Book(1, "The Alchemist", "Paulo Coelho"));

        books.add(new Book(2, "Atomic Habits", "James Clear"));

        books.add(new Book(3, "Think and Grow Rich", "Napoleon Hill"));

        books.add(new Book(4, "1984", "George Orwell"));

        books.add(new Book(5, "To Kill a Mockingbird", "Harper Lee"));

    }

    // Linear Search

    static void linearSearch() {

        sc.nextLine();

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

        String searchTitle = sc.nextLine();

        boolean found = false;

        for (Book book : books) {

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

                System.out.println("Book found:");

                book.display();

                found = true;

            }

        }

        if (!found) {

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

        }

    }

    // Binary Search

    static void binarySearch() {

        sc.nextLine();

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

        String searchTitle = sc.nextLine();

        // Sort books by title before binary search

        books.sort(Comparator.comparing(b -> b.title.toLowerCase()));

        int low = 0, high = books.size() - 1;

        boolean found = false;

        while (low <= high) {

            int mid = (low + high) / 2;

            Book midBook = books.get(mid);

            int compare = searchTitle.compareToIgnoreCase(midBook.title);

            if (compare == 0) {

                System.out.println("Book found:");

                midBook.display();

                found = true;

                break;

            } else if (compare < 0) {

                high = mid - 1;

            } else {

                low = mid + 1;

            }

        }

        if (!found) {

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

        }

    }

    public static void main(String[] args) {

        loadBooks();

        int choice;

        do {

            System.out.println("\nBook Search Menu");

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

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

            System.out.println("3. Exit");

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

            choice = sc.nextInt();

            System.out.println();

            switch (choice) {

                case 1: linearSearch(); break;

                case 2: binarySearch(); break;

                case 3: System.out.println("Exiting..."); break;

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

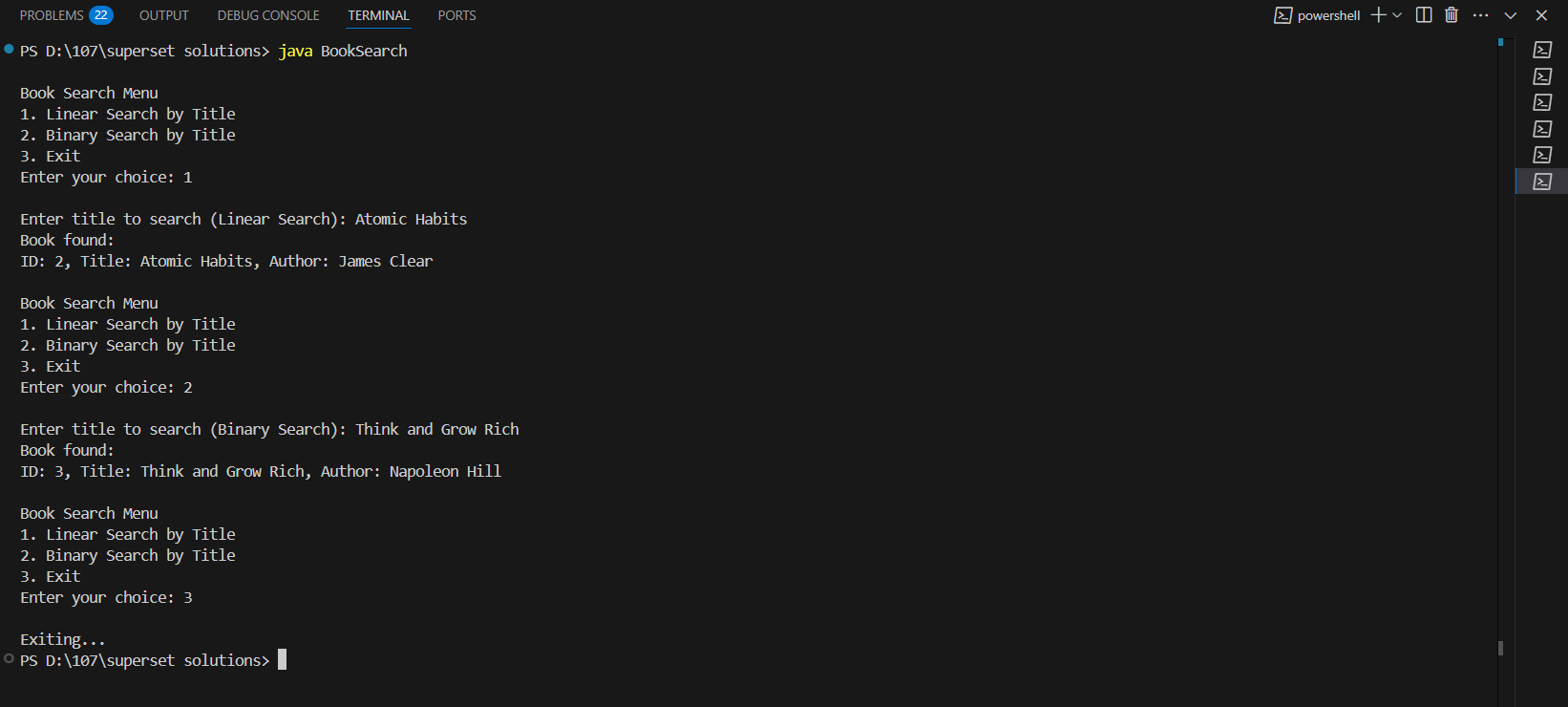
            }

        } while (choice != 3);

    }

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**PROGRAM:**

public class FinancialForecast {

    public static double futureValueRecursive(double initialValue, double growthRate, int years) {

        if (years == 0) {

            return initialValue;

        }

        return (1 + growthRate) \* futureValueRecursive(initialValue, growthRate, years - 1);

    }

    public static double futureValueIterative(double initialValue, double growthRate, int years) {

        double result = initialValue;

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

            result \*= (1 + growthRate);

        }

        return result;

    }

    public static void main(String[] args) {

        double initialValue = 1000.0;

        double growthRate = 0.05;

        int years = 5;

        double recursiveResult = futureValueRecursive(initialValue, growthRate, years);

        double iterativeResult = futureValueIterative(initialValue, growthRate, years);

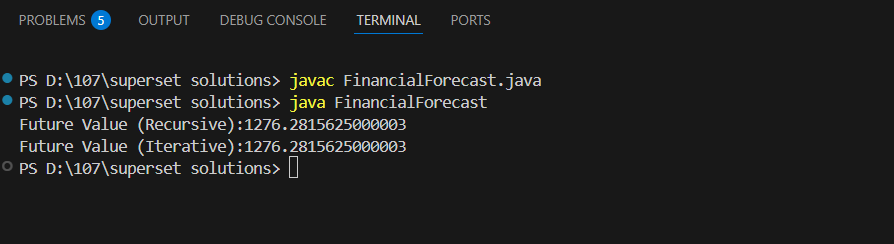
        System.out.println("Future Value (Recursive):" + recursiveResult);

        System.out.println("Future Value (Iterative):" + iterativeResult);

    }

}

**OUTPUT:**

****