Java Data Structures and Algorithms Exercises solution

Exercise 1: Inventory Management System

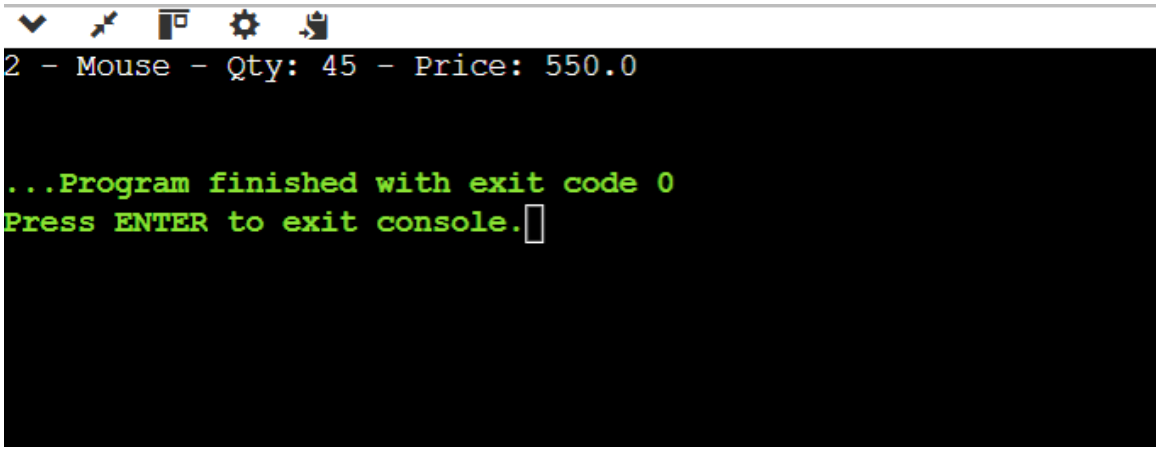
Code :

```
import java.util.*;


class Product {

    int productId;

    String productName;

    int quantity;

    double price;


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

        this.productId = id;

        this.productName = name;

        this.quantity = qty;

        this.price = price;

    }


    @Override

    public String toString() {

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

    }

}


class Inventory {
```

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

void addProduct(Product p) {
    inventory.put(p.productId, p);
}

void updateProduct(int id, int qty, double price) {
    if (inventory.containsKey(id)) {
        Product p = inventory.get(id);
        p.quantity = qty;
        p.price = price;
    }
}

void deleteProduct(int id) {
    inventory.remove(id);
}

void displayInventory() {
    for (Product p : inventory.values()) {
        System.out.println(p);
    }
}

public static void main(String[] args) {
    Inventory inv = new Inventory();
    inv.addProduct(new Product(1, "Laptop", 10, 50000));
    inv.addProduct(new Product(2, "Mouse", 50, 500));
```

```
        inv.updateProduct(2, 45, 550);
        inv.deleteProduct(1);
        inv.displayInventory();
    }
}
```

Output:



Exercise 2: E-commerce Platform Search Function

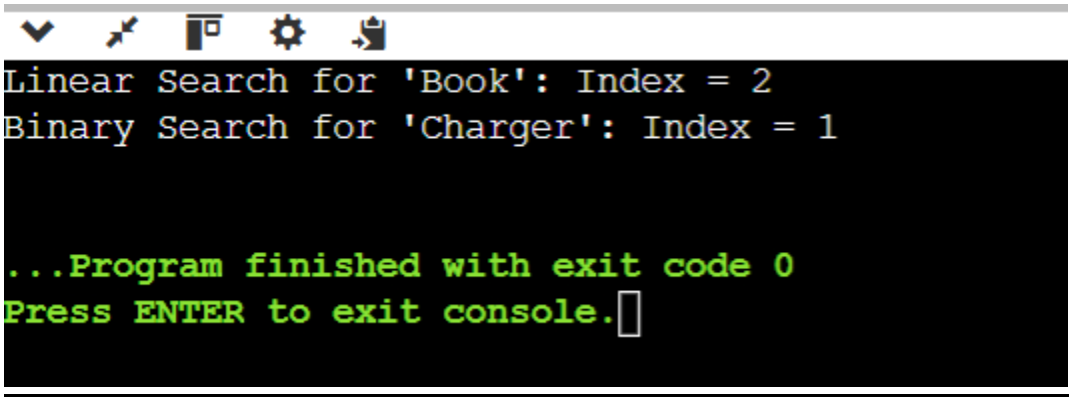
Code :

```
import java.util.Arrays;


class SearchProduct {
    int productId;
    String productName, category;

    public SearchProduct(int id, String name, String category) {
        this.productId = id;
        this.productName = name;
        this.category = category;
    }

    @Override
```

```
    public String toString() {
        return productId + " - " + productName + " - " + category;
    }
}

class Search {
    static int linearSearch(SearchProduct[] arr, String name) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i].productName.equalsIgnoreCase(name)) return i;
        }
        return -1;
    }

    static int binarySearch(SearchProduct[] arr, String name) {
        int l = 0, r = arr.length - 1;
        while (l <= r) {
            int mid = (l + r) / 2;
            int cmp = arr[mid].productName.compareToIgnoreCase(name);
            if (cmp == 0) return mid;
            if (cmp < 0) l = mid + 1;
            else r = mid - 1;
        }
        return -1;
    }
}

public class EcommerceSearchApp {
    public static void main(String[] args) {
```

```
        SearchProduct[] products = {
            new SearchProduct(101, "Phone", "Electronics"),
            new SearchProduct(102, "Charger", "Electronics"),
            new SearchProduct(103, "Book", "Stationery")
        };

        System.out.println("Linear Search for 'Book': Index = " +
Search.linearSearch(products, "Book"));

        Arrays.sort(products, (a, b) ->
a.productName.compareToIgnoreCase(b.productName));
        System.out.println("Binary Search for 'Charger': Index = " +
Search.binarySearch(products, "Charger"));
    }
}
```

Output:



Exercise 3: Sorting Customer Orders

Code:

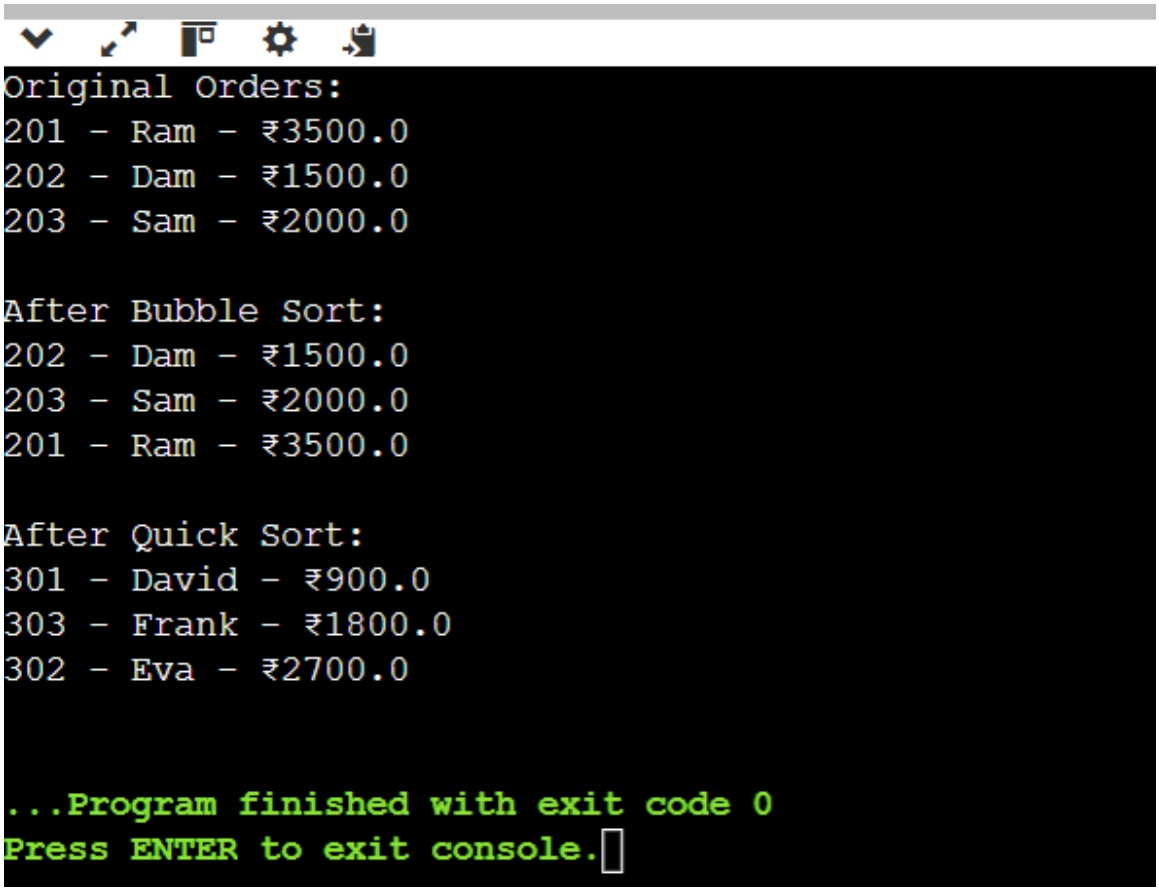
```
class Order {
    int orderId;
    String customerName;
```

```
    double totalPrice;

    public Order(int id, String name, double price) {
        this.orderId = id;
        this.customerName = name;
        this.totalPrice = price;
    }

    @Override
    public String toString() {
        return orderId + " - " + customerName + " - ₹" + totalPrice;
    }
}

class SortOrders {
    static 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;
                }
            }
        }
    }

    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);
    }
}

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 main(String[] args) {
    Order[] orders = {
        new Order(201, "Ram", 3500),
        new Order(202, "Dam", 1500),
```

```
            new Order(203, "Sam", 2000)
        };

        System.out.println("Original Orders:");
        for (Order o : orders) System.out.println(o);

        bubbleSort(orders);
        System.out.println("\nAfter Bubble Sort:");
        for (Order o : orders) System.out.println(o);

        Order[] orders2 = {
            new Order(301, "David", 900),
            new Order(302, "Eva", 2700),
            new Order(303, "Frank", 1800)
        };

        quickSort(orders2, 0, orders2.length - 1);
        System.out.println("\nAfter Quick Sort:");
        for (Order o : orders2) System.out.println(o);
    }
}
```

Output:



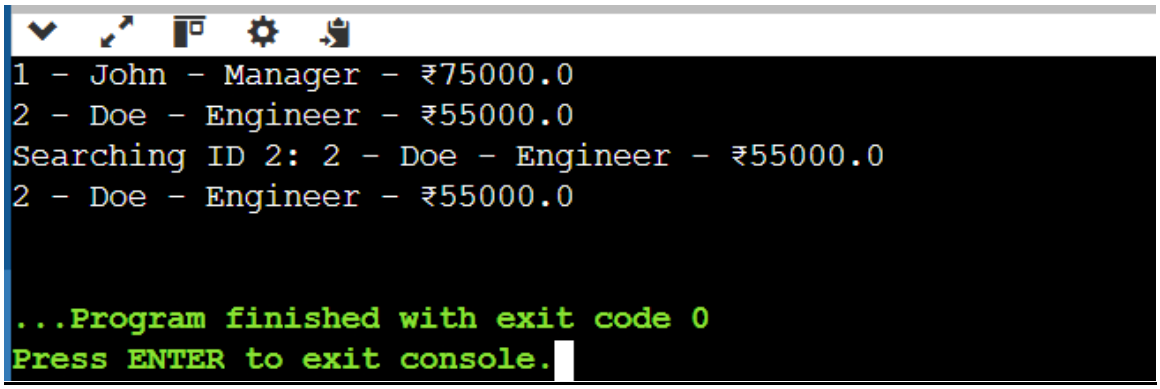
Exercise 4: Employee Management System

```
class Employee {
    int employeeId;
    String name;
    String position;
    double salary;

    public Employee(int id, String name, String pos, double sal) {
        this.employeeId = id;
        this.name = name;
        this.position = pos;
        this.salary = sal;
```

```
    }

    @Override
    public String toString() {
        return employeeId + " - " + name + " - " + position + " - ₹" + salary;
    }
}

public class EmployeeManagementSystem {
    static Employee[] employees = new Employee[100];
    static int count = 0;

    static void addEmployee(Employee e) {
        employees[count++] = e;
    }

    static void displayEmployees() {
        for (int i = 0; i < count; i++) {
            System.out.println(employees[i]);
        }
    }

    static Employee searchEmployee(int id) {
        for (int i = 0; i < count; i++) {
            if (employees[i].employeeId == id)
                return employees[i];
```

```
        }
        return null;
    }

    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];
                }
                count--;
                break;
            }
        }
    }

    public static void main(String[] args) {
        addEmployee(new Employee(1, "John", "Manager", 75000));
        addEmployee(new Employee(2, "Doe", "Engineer", 55000));
        displayEmployees();
        System.out.println("Searching ID 2: " + searchEmployee(2));
        deleteEmployee(1);
        displayEmployees();
    }
}
```

Output:



Exercise 5: Task Management System

Code:

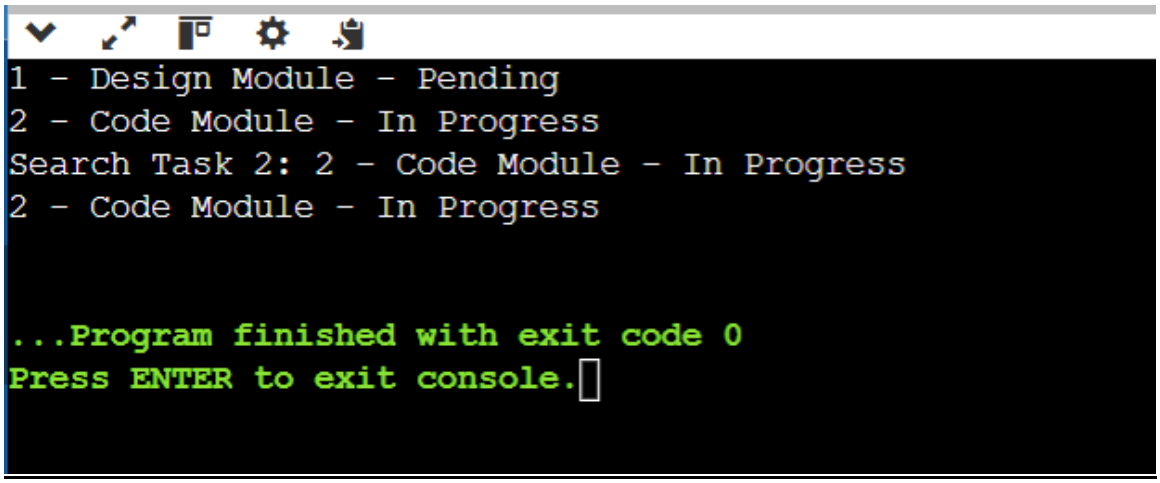
```
class Task {
   int taskId;
   String taskName;
   String status;
   Task next;

   public Task(int id, String name, String status) {
      this.taskId = id;
      this.taskName = name;
      this.status = status;
      this.next = null;
   }

   @Override
   public String toString() {
      return taskId + " - " + taskName + " - " + status;
```

```
    }
}

public class TaskManagementSystem {
    static Task head = null;

    static void addTask(Task newTask) {
        if (head == null) {
            head = newTask;
        } else {
            Task temp = head;
            while (temp.next != null) temp = temp.next;
            temp.next = newTask;
        }
    }

    static void displayTasks() {
        Task temp = head;
        while (temp != null) {
            System.out.println(temp);
            temp = temp.next;
        }
    }

    static Task searchTask(int id) {
```

```
    Task temp = head;
    while (temp != null) {
        if (temp.taskId == id) return temp;
        temp = temp.next;
    }
    return null;
}

static void deleteTask(int id) {
    if (head == null) return;
    if (head.taskId == id) {
        head = head.next;
        return;
    }
    Task temp = head;
    while (temp.next != null && temp.next.taskId != id) {
        temp = temp.next;
    }
    if (temp.next != null) {
        temp.next = temp.next.next;
    }
}

public static void main(String[] args) {
    addTask(new Task(1, "Design Module", "Pending"));
```

```
        addTask(new Task(2, "Code Module", "In Progress"));
        displayTasks();
        System.out.println("Search Task 2: " + searchTask(2));
        deleteTask(1);
        displayTasks();
    }
}
```

Output:



Exercise 6: Library Management System

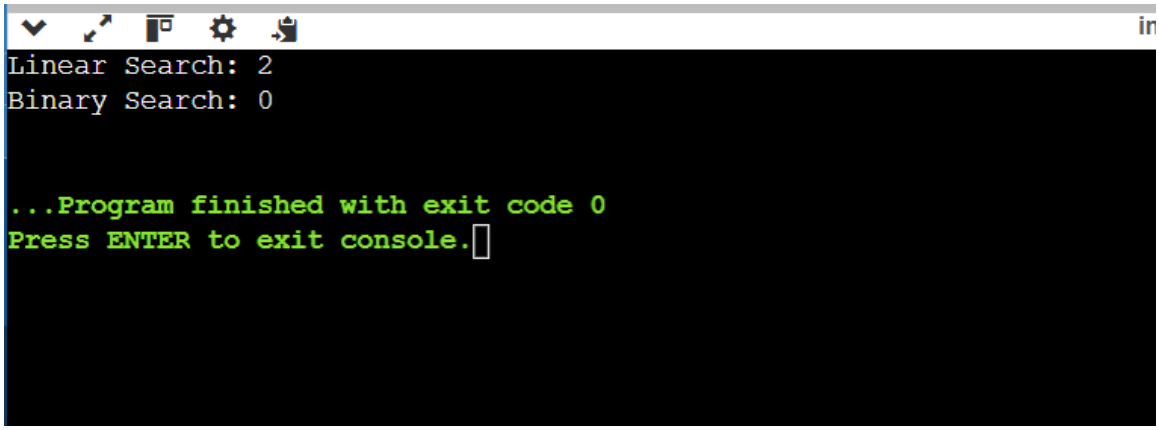
Code:

```
import java.util.Arrays;


class Book {
    int bookId;
    String title;
    String author;

    public Book(int id, String title, String author) {
```

```
        this.bookId = id;
        this.title = title;
        this.author = author;
    }

    @Override
    public String toString() {
        return bookId + " - " + title + " - " + author;
    }
}

public class LibraryManagementSystem {
    static Book[] books = {
        new Book(1, "Algorithms", "Cormen"),
        new Book(2, "Data Structures", "Lafore"),
        new Book(3, "Java Basics", "Gosling")
    };

    static int linearSearch(String title) {
        for (int i = 0; i < books.length; i++) {
            if (books[i].title.equalsIgnoreCase(title)) return i;
        }
        return -1;
    }
```

```
    static int binarySearch(String title) {
        Arrays.sort(books, (a, b) -> a.title.compareToIgnoreCase(b.title));
        int l = 0, r = books.length - 1;
        while (l <= r) {
            int mid = (l + r) / 2;
            int cmp = books[mid].title.compareToIgnoreCase(title);
            if (cmp == 0) return mid;
            if (cmp < 0) l = mid + 1;
            else r = mid - 1;
        }
        return -1;
    }

    public static void main(String[] args) {
        System.out.println("Linear Search: " + linearSearch("Java Basics"));
        System.out.println("Binary Search: " + binarySearch("Algorithms"));
    }
}
```

Output:

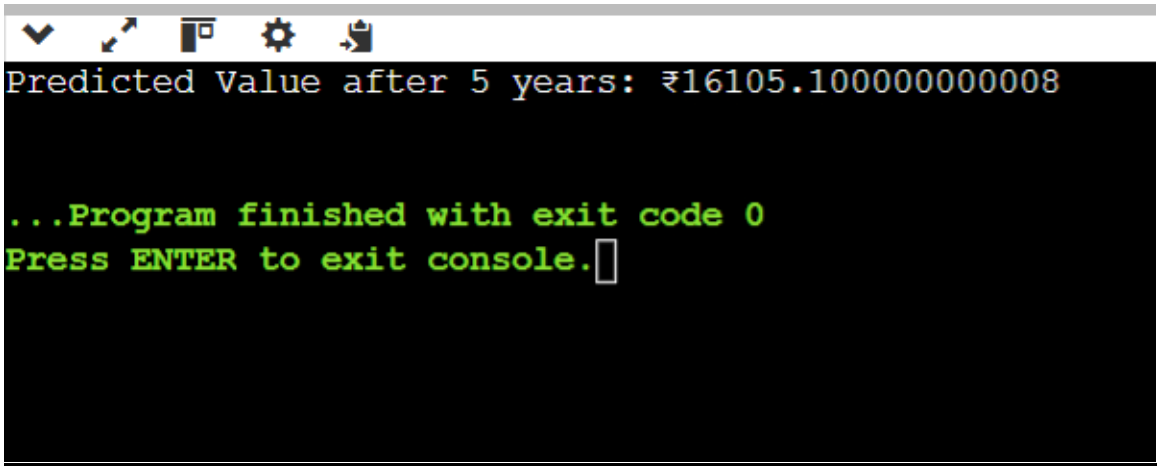


Exercise 7: Financial Forecasting

Code:

```
public class FinancialForecasting {
    static double predictValue(int years, double currentValue, double rate) {
        if (years == 0) return currentValue;
        return predictValue(years - 1, currentValue * (1 + rate), rate);
    }

    public static void main(String[] args) {
        double futureValue = predictValue(5, 10000, 0.10);
        System.out.println("Predicted Value after 5 years: ₹" + futureValue);
    }
}
```

Output:



Submitted By:

Name :Lingaraj Nayak

Superset ID:6387607