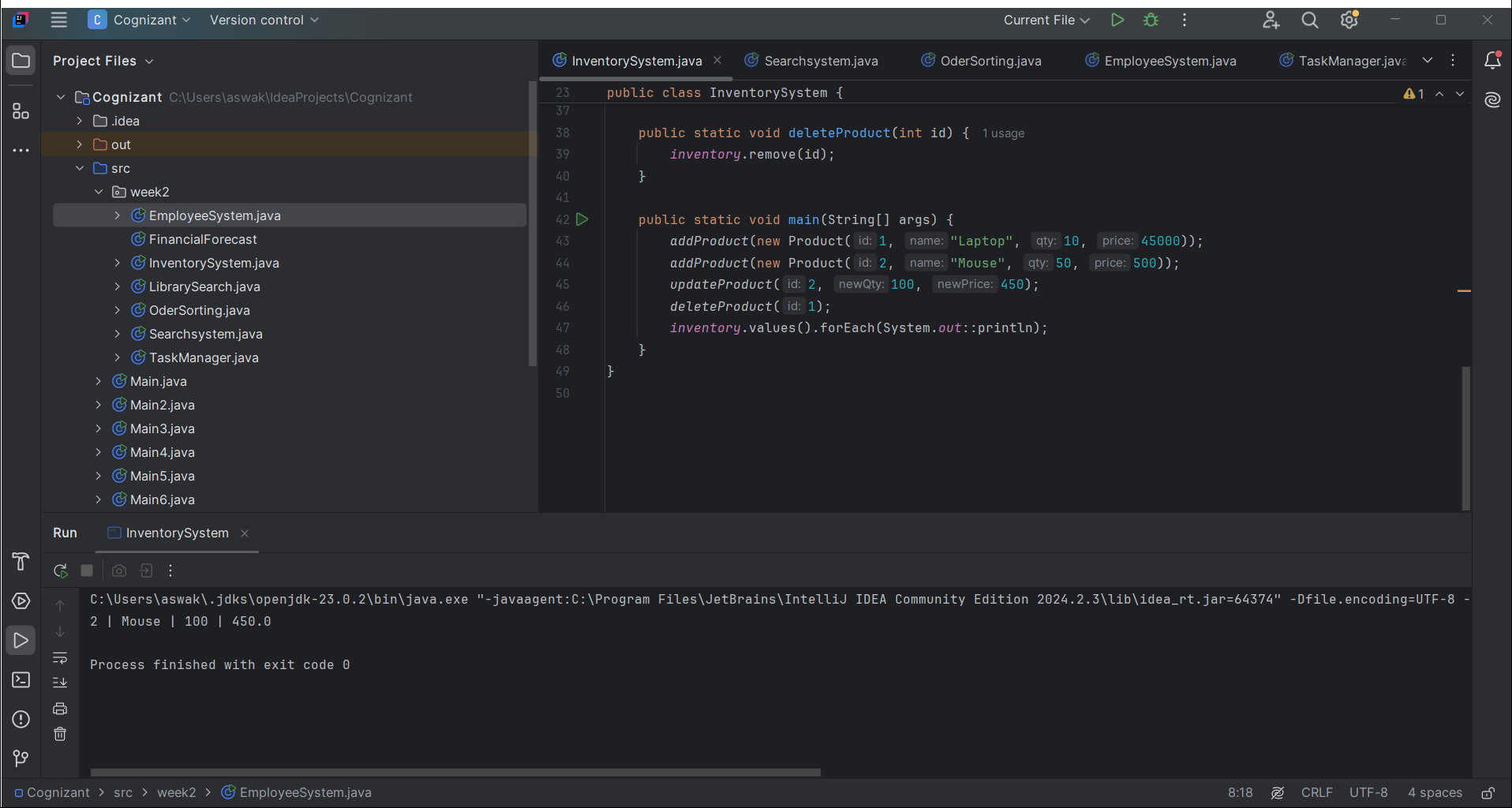
Data Structures & Algorithms Exercises - IntelliJ Ready Java Code

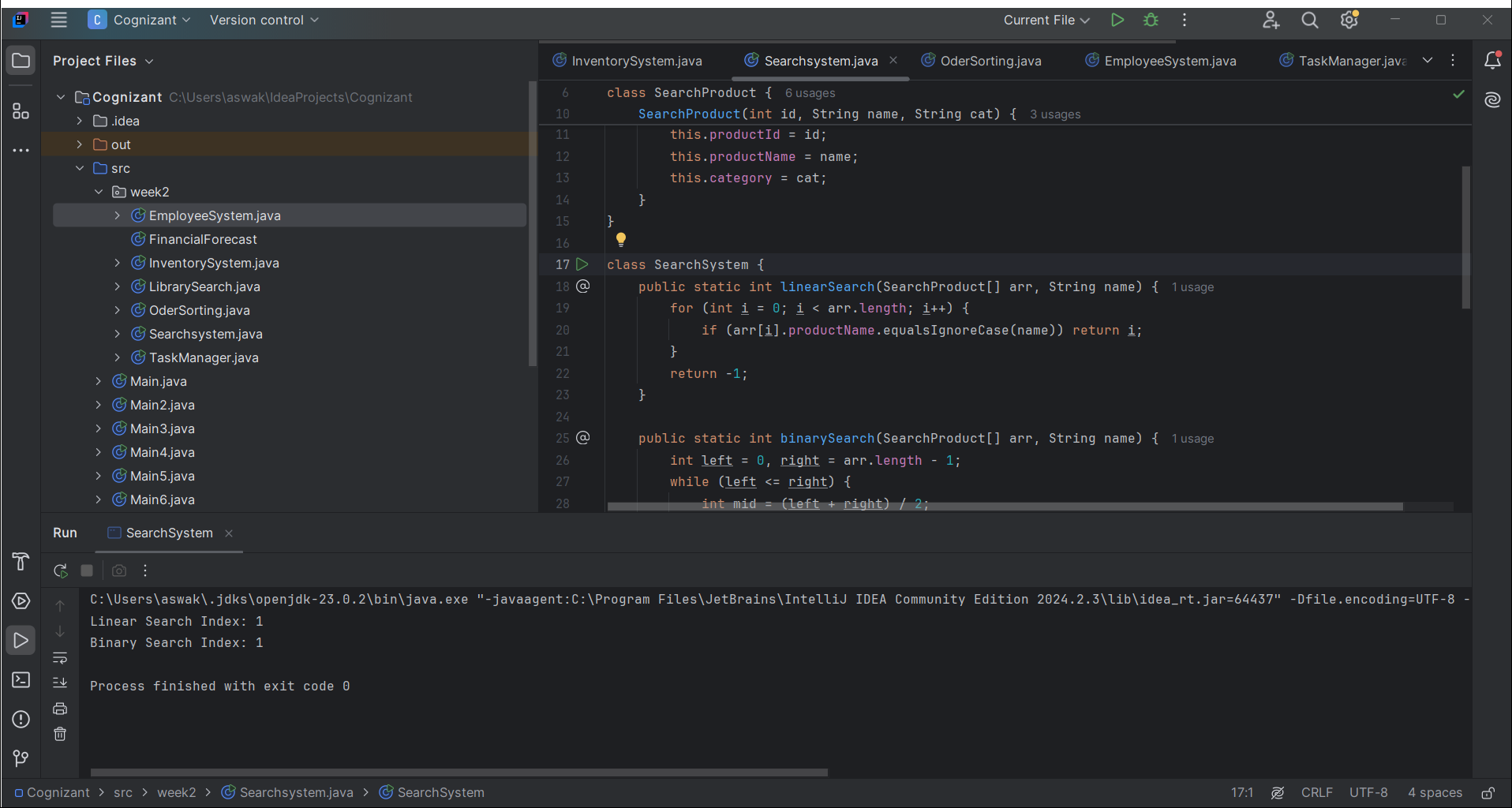
# Exercise 1: Inventory Management System

import java.util.HashMap;  
import java.util.Map;  
  
class Product {  
 int productId;  
 String productName;  
 int quantity;  
 double price;  
  
 Product(int id, String name, int qty, double price) {  
 this.productId = id;  
 this.productName = name;  
 this.quantity = qty;  
 this.price = price;  
 }  
  
 public String toString() {  
 return productId + " | " + productName + " | " + quantity + " | " + price;  
 }  
}  
  
public class InventorySystem {  
 static Map<Integer, Product> inventory = new HashMap<>();  
  
 public static void addProduct(Product p) {  
 inventory.put(p.productId, p);  
 }  
  
 public static void updateProduct(int id, int newQty, double newPrice) {  
 Product p = inventory.get(id);  
 if (p != null) {  
 p.quantity = newQty;  
 p.price = newPrice;  
 }  
 }  
  
 public static void deleteProduct(int id) {  
 inventory.remove(id);  
 }  
  
 public static void main(String[] args) {  
 addProduct(new Product(1, "Laptop", 10, 45000));  
 addProduct(new Product(2, "Mouse", 50, 500));  
 updateProduct(2, 100, 450);  
 deleteProduct(1);  
 inventory.values().forEach(System.out::println);  
 }  
}



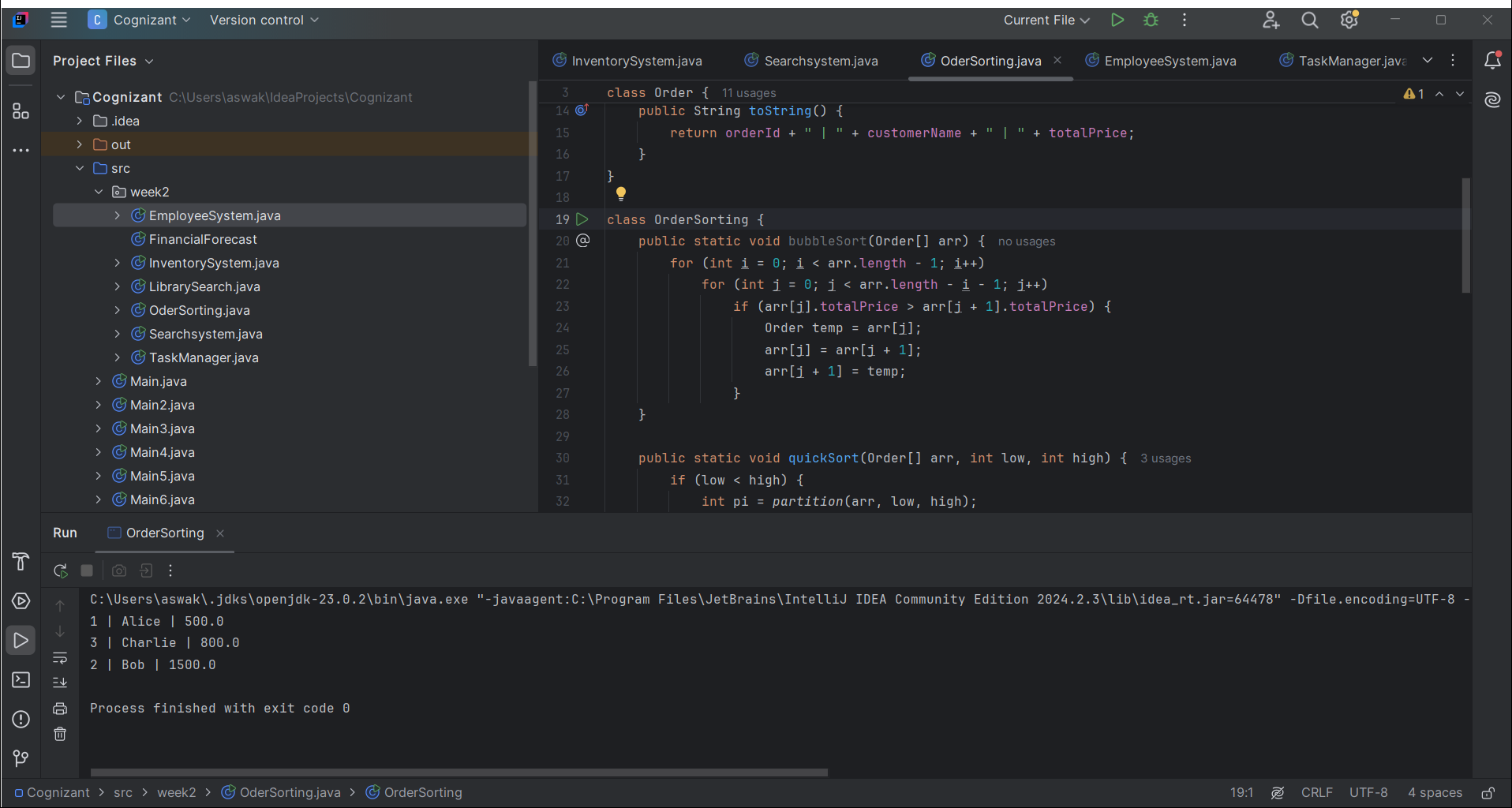
# Exercise 2: E-commerce Platform Search

class Product {  
 int productId;  
 String productName, category;  
  
 Product(int id, String name, String cat) {  
 this.productId = id;  
 this.productName = name;  
 this.category = cat;  
 }  
}  
  
public class SearchSystem {  
 public static int linearSearch(Product[] arr, String name) {  
 for (int i = 0; i < arr.length; i++) {  
 if (arr[i].productName.equals(name)) return i;  
 }  
 return -1;  
 }  
  
 public static int binarySearch(Product[] arr, String name) {  
 int left = 0, right = arr.length - 1;  
 while (left <= right) {  
 int mid = (left + right) / 2;  
 int cmp = arr[mid].productName.compareTo(name);  
 if (cmp == 0) return mid;  
 else if (cmp < 0) left = mid + 1;  
 else right = mid - 1;  
 }  
 return -1;  
 }  
  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(1, "Camera", "Electronics"),  
 new Product(2, "Laptop", "Electronics"),  
 new Product(3, "Phone", "Electronics")  
 };  
 System.out.println("Linear Search Index: " + linearSearch(products, "Laptop"));  
 System.out.println("Binary Search Index: " + binarySearch(products, "Laptop"));  
 }  
}

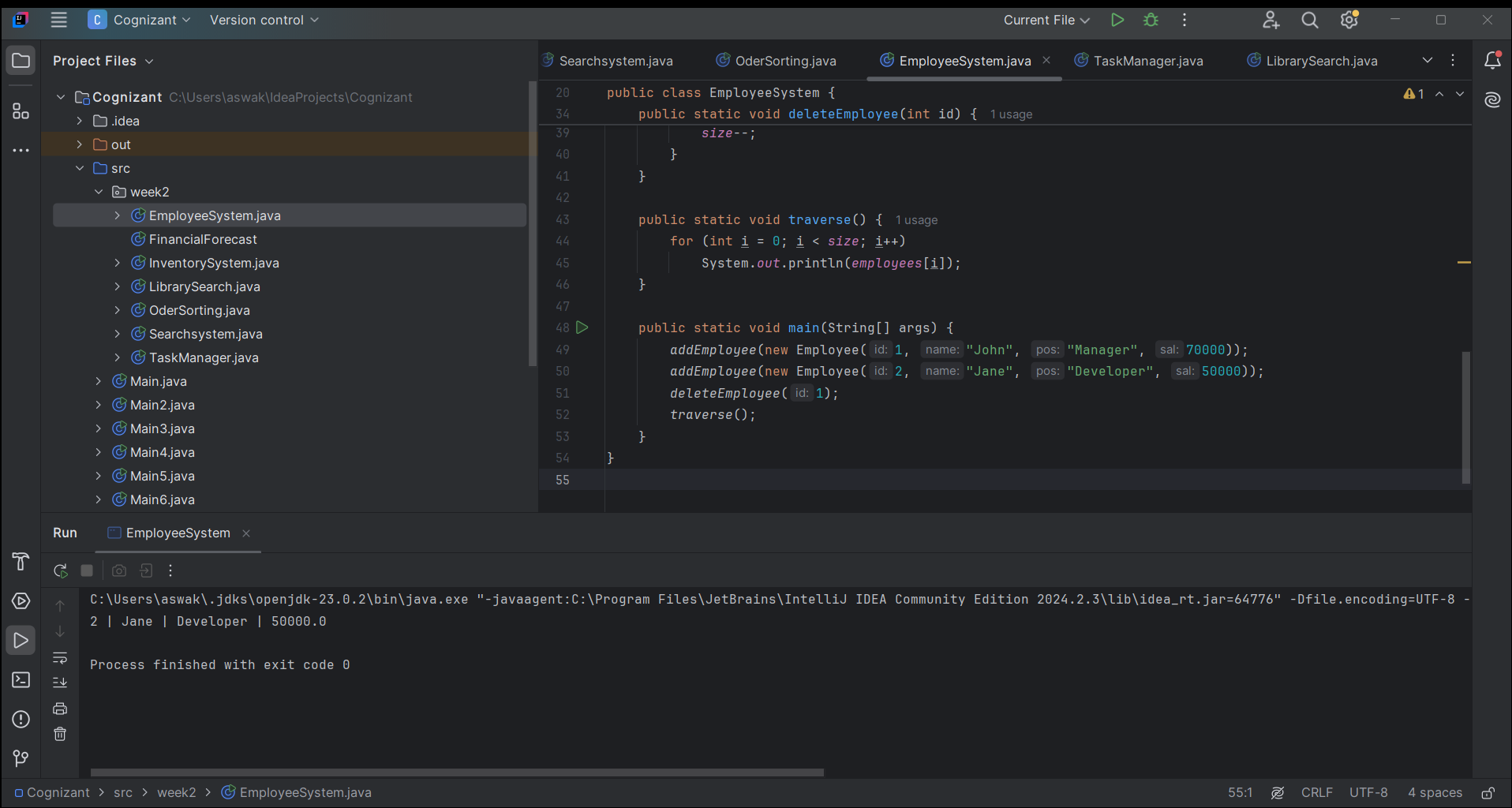


# Exercise 3: Sorting Customer Orders

class Order {  
 int orderId;  
 String customerName;  
 double totalPrice;  
  
 Order(int id, String name, double price) {  
 this.orderId = id;  
 this.customerName = name;  
 this.totalPrice = price;  
 }  
  
 public String toString() {  
 return orderId + " | " + customerName + " | " + totalPrice;  
 }  
}  
  
public class OrderSorting {  
 public static void bubbleSort(Order[] arr) {  
 for (int i = 0; i < arr.length - 1; i++)  
 for (int j = 0; j < arr.length - i - 1; j++)  
 if (arr[j].totalPrice > arr[j + 1].totalPrice) {  
 Order temp = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = temp;  
 }  
 }  
  
 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);  
 }  
 }  
  
 private 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 main(String[] args) {  
 Order[] orders = {  
 new Order(1, "Alice", 500),  
 new Order(2, "Bob", 1500),  
 new Order(3, "Charlie", 800)  
 };  
 quickSort(orders, 0, orders.length - 1);  
 for (Order o : orders) System.out.println(o);  
 }  
}

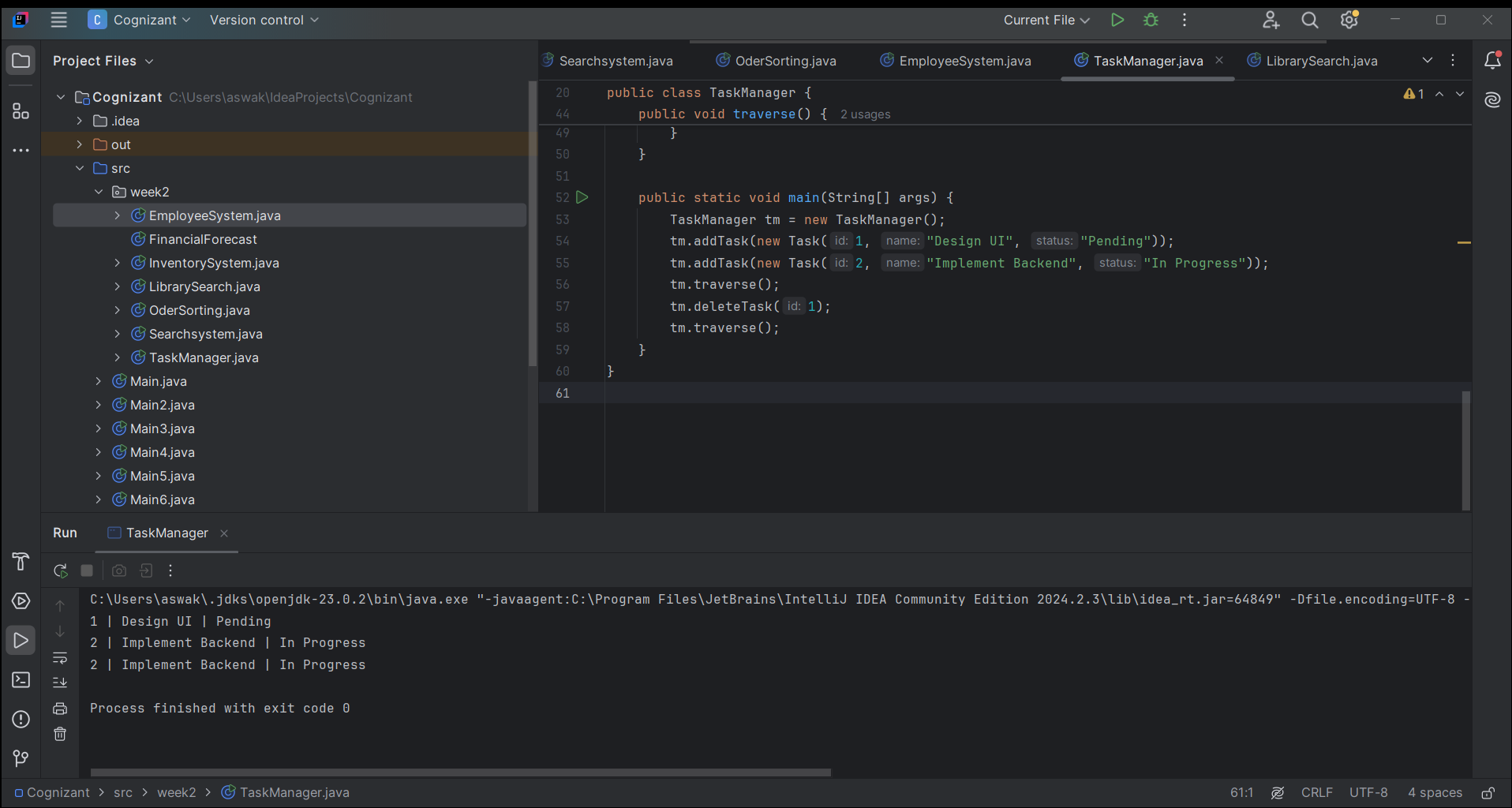


# Exercise 4: Employee Management System

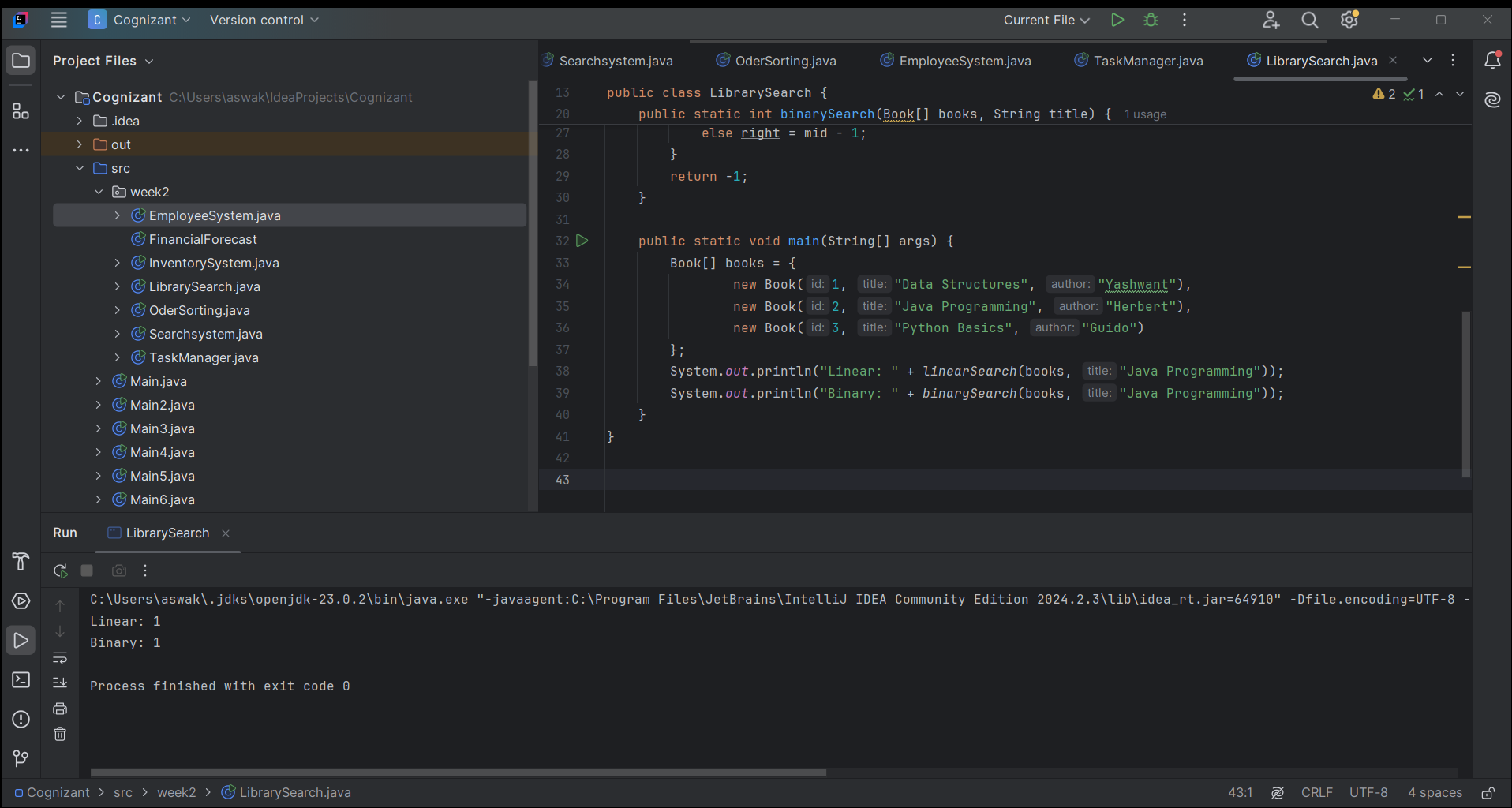
class Employee {  
 int employeeId;  
 String name, position;  
 double salary;  
  
 Employee(int id, String name, String pos, double sal) {  
 this.employeeId = id;  
 this.name = name;  
 this.position = pos;  
 this.salary = sal;  
 }  
  
 public String toString() {  
 return employeeId + " | " + name + " | " + position + " | " + salary;  
 }  
}  
  
public class EmployeeSystem {  
 static Employee[] employees = new Employee[10];  
 static int size = 0;  
  
 public static void addEmployee(Employee e) {  
 employees[size++] = e;  
 }  
  
 public static int searchEmployee(int id) {  
 for (int i = 0; i < size; i++)  
 if (employees[i].employeeId == id) return i;  
 return -1;  
 }  
  
 public static void deleteEmployee(int id) {  
 int index = searchEmployee(id);  
 if (index != -1) {  
 for (int i = index; i < size - 1; i++)  
 employees[i] = employees[i + 1];  
 size--;  
 }  
 }  
  
 public static void traverse() {  
 for (int i = 0; i < size; i++)  
 System.out.println(employees[i]);  
 }  
  
 public static void main(String[] args) {  
 addEmployee(new Employee(1, "John", "Manager", 70000));  
 addEmployee(new Employee(2, "Jane", "Developer", 50000));  
 deleteEmployee(1);  
 traverse();  
 }  
}  


# Exercise 5: Task Management System

class Task {  
 int taskId;  
 String taskName;  
 String status;  
 Task next;  
  
 Task(int id, String name, String status) {  
 this.taskId = id;  
 this.taskName = name;  
 this.status = status;  
 }  
  
 public String toString() {  
 return taskId + " | " + taskName + " | " + status;  
 }  
}  
  
public class TaskManager {  
 Task head;  
  
 public void addTask(Task newTask) {  
 if (head == null) head = newTask;  
 else {  
 Task temp = head;  
 while (temp.next != null) temp = temp.next;  
 temp.next = newTask;  
 }  
 }  
  
 public 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 void traverse() {  
 Task temp = head;  
 while (temp != null) {  
 System.out.println(temp);  
 temp = temp.next;  
 }  
 }  
  
 public static void main(String[] args) {  
 TaskManager tm = new TaskManager();  
 tm.addTask(new Task(1, "Design UI", "Pending"));  
 tm.addTask(new Task(2, "Implement Backend", "In Progress"));  
 tm.traverse();  
 tm.deleteTask(1);  
 tm.traverse();  
 }  
}



# Exercise 6: Library Management Search

class Book {  
 int bookId;  
 String title, author;  
  
 Book(int id, String title, String author) {  
 this.bookId = id;  
 this.title = title;  
 this.author = author;  
 }  
}  
  
public class LibrarySearch {  
 public static int linearSearch(Book[] books, String title) {  
 for (int i = 0; i < books.length; i++)  
 if (books[i].title.equals(title)) return i;  
 return -1;  
 }  
  
 public static int binarySearch(Book[] books, String title) {  
 int left = 0, right = books.length - 1;  
 while (left <= right) {  
 int mid = (left + right) / 2;  
 int cmp = books[mid].title.compareTo(title);  
 if (cmp == 0) return mid;  
 else if (cmp < 0) left = mid + 1;  
 else right = mid - 1;  
 }  
 return -1;  
 }  
  
 public static void main(String[] args) {  
 Book[] books = {  
 new Book(1, "Data Structures", "Yashwant"),  
 new Book(2, "Java Programming", "Herbert"),  
 new Book(3, "Python Basics", "Guido")  
 };  
 System.out.println("Linear: " + linearSearch(books, "Java Programming"));  
 System.out.println("Binary: " + binarySearch(books, "Java Programming"));  
 }  
}  


# Exercise 7: Financial Forecasting

public class FinancialForecast {  
 public static double predictValue(double currentValue, double growthRate, int years) {  
 if (years == 0) return currentValue;  
 return predictValue(currentValue \* (1 + growthRate), growthRate, years - 1);  
 }  
  
 public static void main(String[] args) {  
 double future = predictValue(10000, 0.10, 5);  
 System.out.println("Future Value after 5 years: " + future);  
 }  
}  
