**Algorithms-Data Structures**

**Exercise 1: Inventory Management System**

Source code:

import java.util.HashMap;

import java.util.Scanner;

public class InventoryManagementSystem {

static class Product {

String productId;

String productName;

int quantity;

double price;

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

public String toString() {

return "[" + productId + "] " + productName + " - Qty: " + quantity + ", ₹" + price;

}

}

static class Inventory {

private HashMap<String, Product> products = new HashMap<>();

public void addProduct(Product product) {

products.put(product.productId, product);

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

}

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

Product product = products.get(productId);

if (product != null) {

product.quantity = quantity;

product.price = price;

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

} else {

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

}

}

public void deleteProduct(String productId) {

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

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

} else {

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

}

}

public void displayAll() {

if (products.isEmpty()) {

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

} else {

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

System.out.println(p);

}

}

}

}

public static void main(String[] args) {

Inventory inventory = new Inventory();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\n1. Add Product\n2. Update Product\n3. Delete Product\n4. Display All\n5. Exit");

int choice = scanner.nextInt();

scanner.nextLine(); // clear input

if (choice == 1) {

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

String id = scanner.nextLine();

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

String name = scanner.nextLine();

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

int qty = scanner.nextInt();

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

double price = scanner.nextDouble();

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

} else if (choice == 2) {

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

String id = scanner.nextLine();

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

int qty = scanner.nextInt();

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

double price = scanner.nextDouble();

inventory.updateProduct(id, qty, price);

} else if (choice == 3) {

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

String id = scanner.nextLine();

inventory.deleteProduct(id);

} else if (choice == 4) {

inventory.displayAll();

} else if (choice == 5) {

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

break;

} else {

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

}

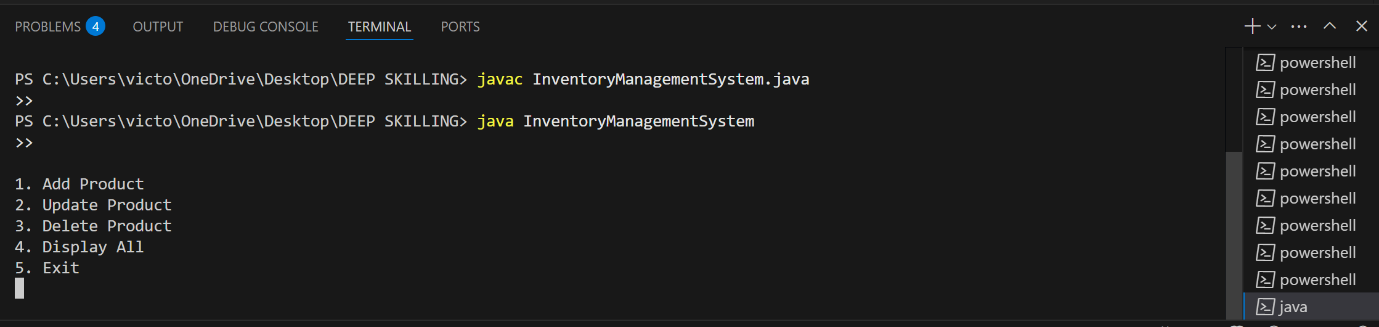
}

scanner.close();

}

}

Output:



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

Source code:

import java.util.Arrays;

import java.util.Comparator;

public class EcommerceSearchExample {

static 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 toString() {

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

}

}

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

for (int i = 0; i < products.length; i++) {

if (products[i].productName.equalsIgnoreCase(targetName)) {

return i;

}

}

return -1;

}

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

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

if (compare == 0) return mid;

if (compare < 0) low = mid + 1;

else high = mid - 1;

}

return -1;

}

public static void main(String[] args) {

Product[] products = {

new Product("P101", "Shoes", "Footwear"),

new Product("P102", "Laptop", "Electronics"),

new Product("P103", "Watch", "Accessories"),

new Product("P104", "Mobile", "Electronics"),

new Product("P105", "Bag", "Travel")

};

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

int index1 = linearSearch(products, "Watch");

System.out.println(index1 != -1 ? "Found: " + products[index1] : "Not found");

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

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

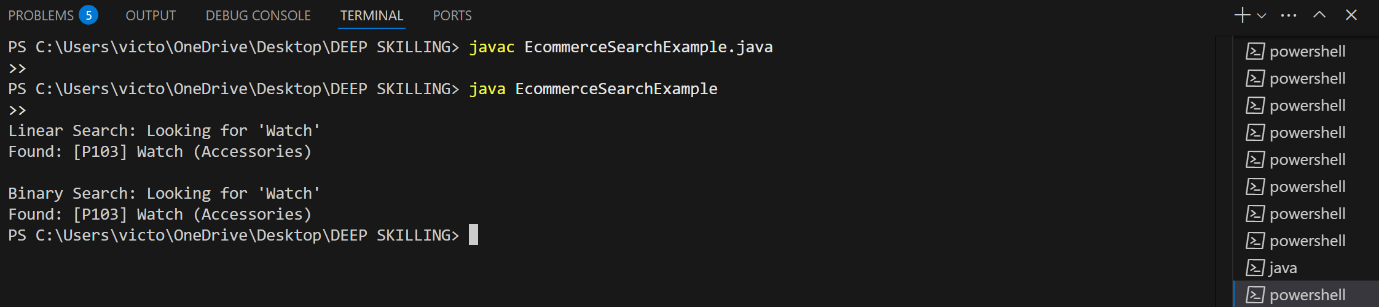
int index2 = binarySearch(products, "Watch");

System.out.println(index2 != -1 ? "Found: " + products[index2] : "Not found");

}

}

Output:

****

**Exercise 3: Sorting Customer Orders**

Source code:

public class OrderSortingExample {

static class Order {

String orderId;

String customerName;

double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public String toString() {

return "[" + orderId + "] " + customerName + " - ₹" + totalPrice;

}

}

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;

}

}

}

}

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;

}

static void printOrders(Order[] orders) {

for (Order order : orders) {

System.out.println(order);

}

}

public static void main(String[] args) {

Order[] orders = {

new Order("O101", "Alice", 1500.0),

new Order("O102", "Bob", 3000.0),

new Order("O103", "Charlie", 1200.0),

new Order("O104", "Diana", 8000.0),

new Order("O105", "Eve", 500.0)

};

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

printOrders(orders);

System.out.println("\nBubble Sort by totalPrice:");

bubbleSort(orders);

printOrders(orders);

Order[] orders2 = {

new Order("O101", "Alice", 1500.0),

new Order("O102", "Bob", 3000.0),

new Order("O103", "Charlie", 1200.0),

new Order("O104", "Diana", 8000.0),

new Order("O105", "Eve", 500.0)

};

System.out.println("\nQuick Sort by totalPrice:");

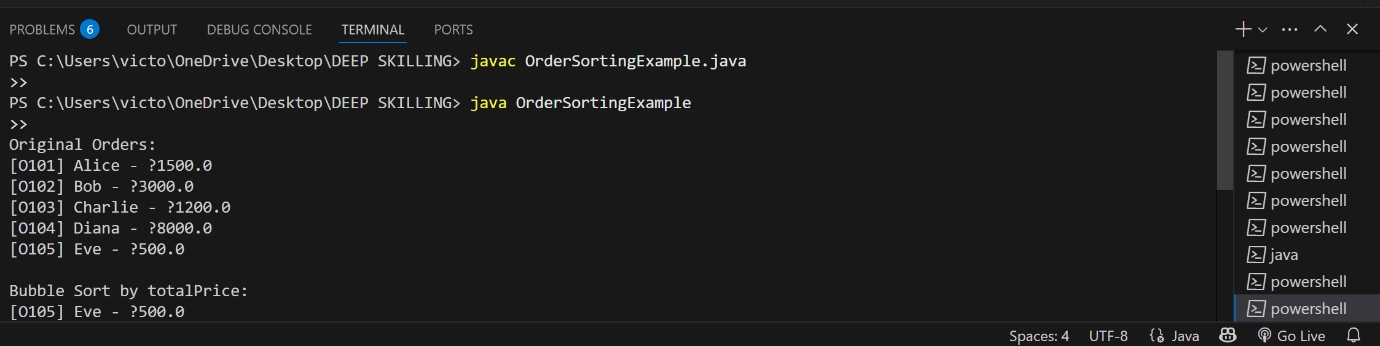
quickSort(orders2, 0, orders2.length - 1);

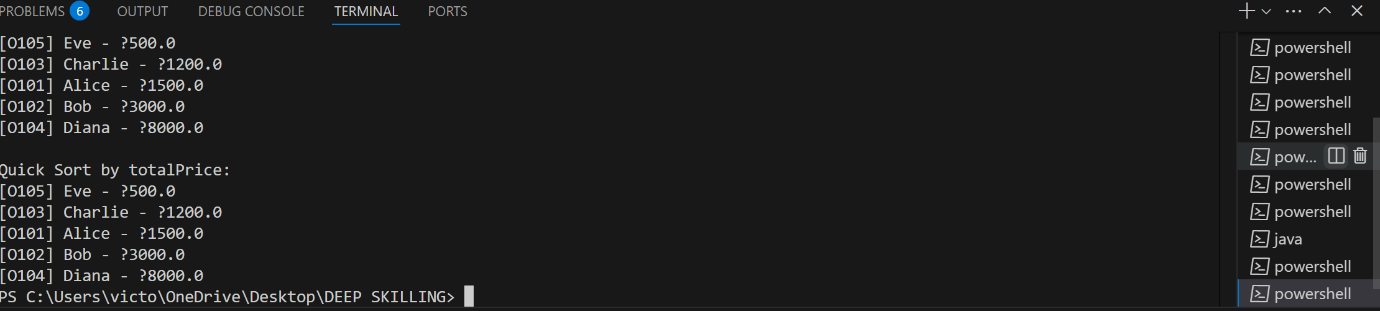
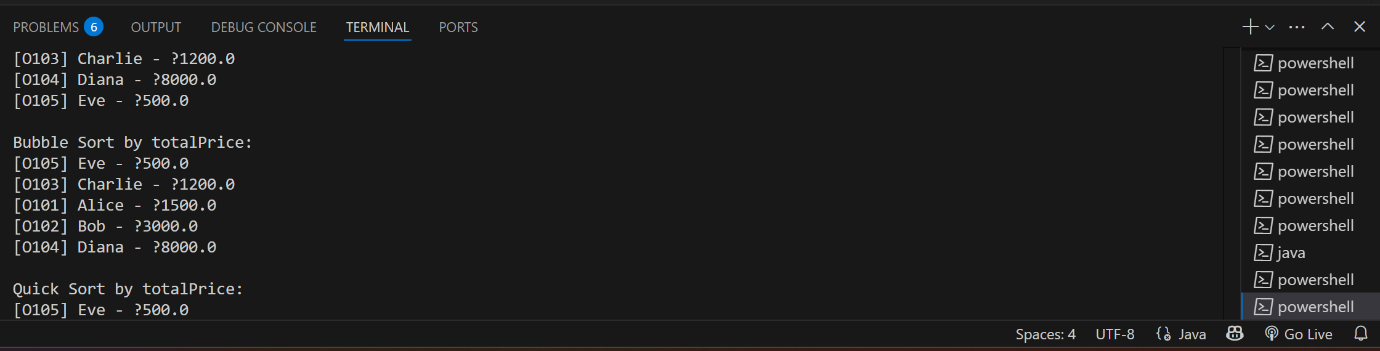
printOrders(orders2);

}

}

Output:





**Exercise 4: Employee Management System**

Souce code:

import java.util.Scanner;

public class EmployeeManagementSystem {

static 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 String toString() {

return "[" + employeeId + "] " + name + " - " + position + " - ₹" + salary;

}

}

static class EmployeeManager {

private Employee[] employees;

private int size;

public EmployeeManager(int capacity) {

employees = new Employee[capacity];

size = 0;

}

public void addEmployee(Employee e) {

if (size < employees.length) {

employees[size++] = e;

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

} else {

System.out.println("Array full. Cannot add more employees.");

}

}

public void searchEmployee(int id) {

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

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

System.out.println("Found: " + employees[i]);

return;

}

}

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

}

public void deleteEmployee(int id) {

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

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

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

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

}

employees[--size] = null;

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

return;

}

}

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

}

public void traverseEmployees() {

if (size == 0) {

System.out.println("No employees to show.");

return;

}

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

System.out.println(employees[i]);

}

}

}

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager(100);

Scanner sc = new Scanner(System.in);

while (true) {

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

int choice = sc.nextInt();

sc.nextLine();

if (choice == 1) {

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

int id = sc.nextInt();

sc.nextLine();

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

String name = sc.nextLine();

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

String pos = sc.nextLine();

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

double sal = sc.nextDouble();

manager.addEmployee(new Employee(id, name, pos, sal));

} else if (choice == 2) {

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

int id = sc.nextInt();

manager.searchEmployee(id);

} else if (choice == 3) {

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

int id = sc.nextInt();

manager.deleteEmployee(id);

} else if (choice == 4) {

manager.traverseEmployees();

} else if (choice == 5) {

break;

} else {

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

}

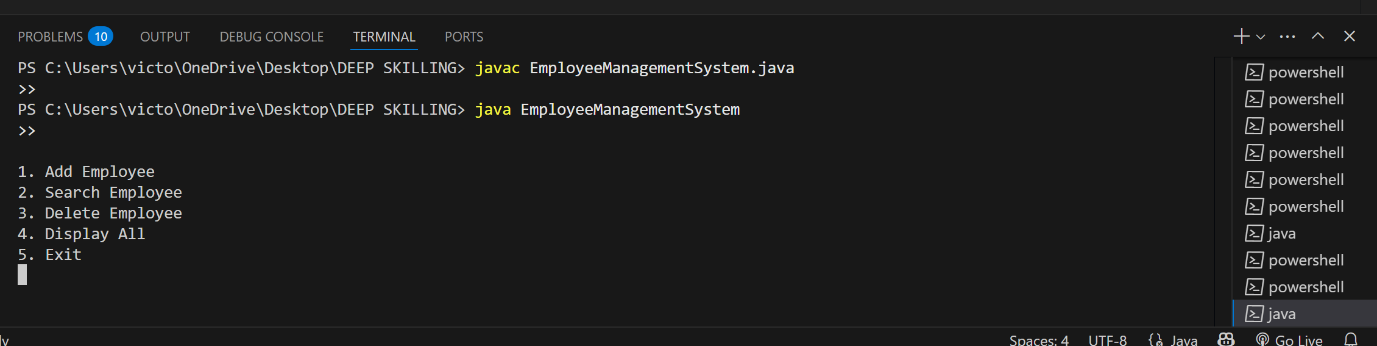
}

sc.close();

}

}

Output:



**Exercise 5: Task Management System**

Source code:

public class TaskManagementSystem {

static class Task {

int taskId;

String taskName;

String status;

Task next;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

public String toString() {

return "[" + taskId + "] " + taskName + " - " + status;

}

}

static class TaskManager {

private Task head;

public void addTask(int taskId, String taskName, String status) {

Task newTask = new Task(taskId, taskName, status);

if (head == null) {

head = newTask;

} else {

Task current = head;

while (current.next != null) {

current = current.next;

}

current.next = newTask;

}

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

}

public void traverseTasks() {

if (head == null) {

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

return;

}

Task current = head;

while (current != null) {

System.out.println(current);

current = current.next;

}

}

public void searchTask(int taskId) {

Task current = head;

while (current != null) {

if (current.taskId == taskId) {

System.out.println("Found: " + current);

return;

}

current = current.next;

}

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

}

public void deleteTask(int taskId) {

if (head == null) {

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

return;

}

if (head.taskId == taskId) {

head = head.next;

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

return;

}

Task current = head;

while (current.next != null) {

if (current.next.taskId == taskId) {

current.next = current.next.next;

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

return;

}

current = current.next;

}

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

}

}

public static void main(String[] args) {

TaskManager manager = new TaskManager();

manager.addTask(1, "Design UI", "Pending");

manager.addTask(2, "Write Backend", "In Progress");

manager.addTask(3, "Test Application", "Not Started");

System.out.println("\nAll Tasks:");

manager.traverseTasks();

System.out.println("\nSearching for Task 2:");

manager.searchTask(2);

System.out.println("\nDeleting Task 2:");

manager.deleteTask(2);

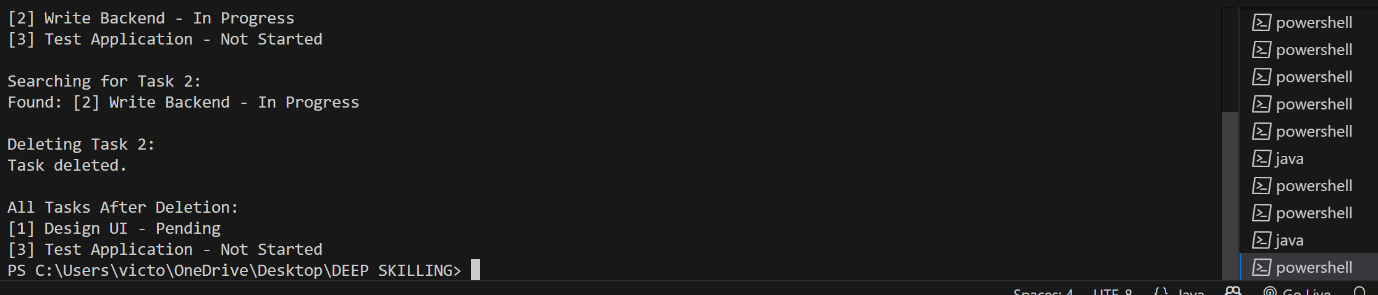
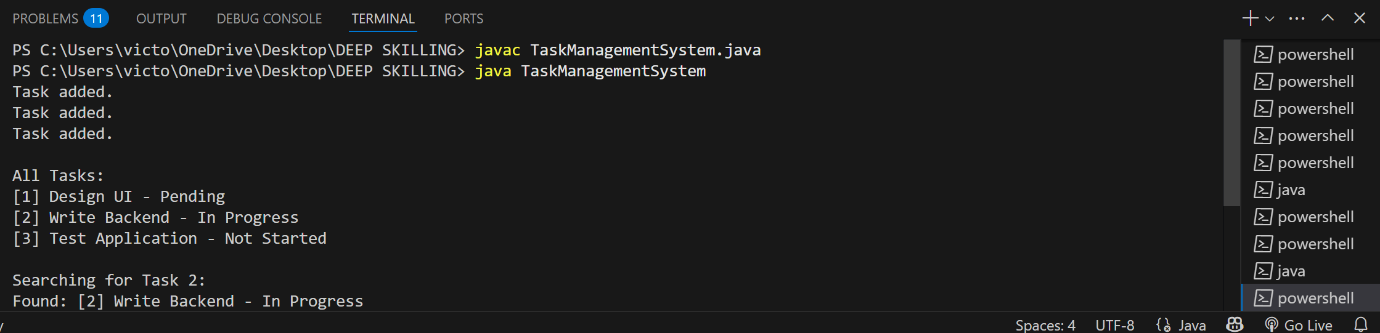
System.out.println("\nAll Tasks After Deletion:");

manager.traverseTasks();

}

}

Output:



**Exercise 6: Library Management System**

Source code:

import java.util.Arrays;

import java.util.Comparator;

public class LibraryManagementSystem {

static 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 String toString() {

return "[" + bookId + "] " + title + " by " + author;

}

}

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

for (int i = 0; i < books.length; i++) {

if (books[i].title.equalsIgnoreCase(targetTitle)) {

return i;

}

}

return -1;

}

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

int low = 0, high = books.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

if (result == 0) return mid;

else if (result < 0) low = mid + 1;

else high = mid - 1;

}

return -1;

}

static void printBooks(Book[] books) {

for (Book book : books) {

System.out.println(book);

}

}

public static void main(String[] args) {

Book[] books = {

new Book(101, "Java Programming", "James Gosling"),

new Book(102, "Data Structures", "Robert Lafore"),

new Book(103, "Algorithms", "Thomas Cormen"),

new Book(104, "Operating Systems", "Silberschatz"),

new Book(105, "Database Systems", "Elmasri")

};

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

int index1 = linearSearch(books, "Algorithms");

System.out.println(index1 != -1 ? "Found: " + books[index1] : "Not found");

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

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

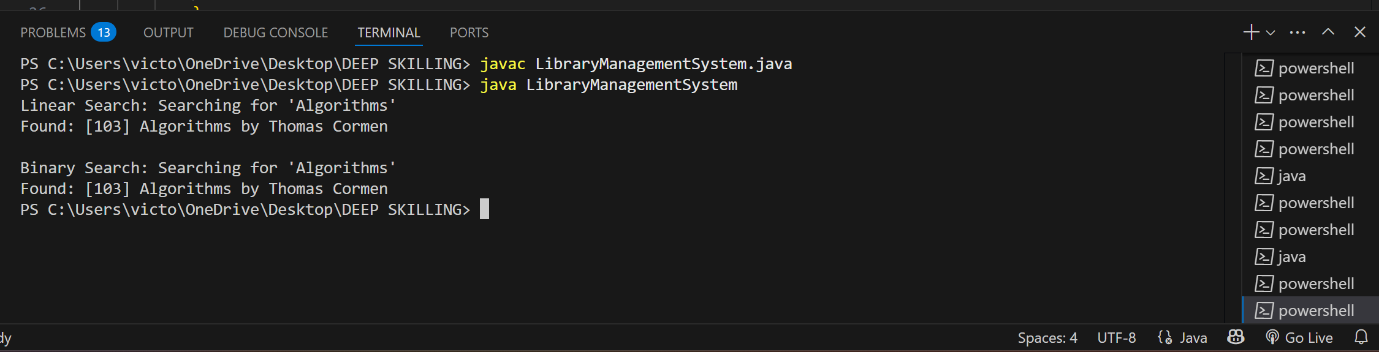
int index2 = binarySearch(books, "Algorithms");

System.out.println(index2 != -1 ? "Found: " + books[index2] : "Not found");

}

}

Output:



**Exercise 7: Financial Forecasting**

Source code:

import java.util.HashMap;

public class FinancialForecasting {

static double baseValue = 10000.0; // Starting value

static double annualGrowthRate = 0.10; // 10% growth

// Recursive method

public static double forecast(int year) {

if (year == 0) return baseValue;

return forecast(year - 1) \* (1 + annualGrowthRate);

}

// Optimized with memoization

static HashMap<Integer, Double> memo = new HashMap<>();

public static double forecastMemo(int year) {

if (year == 0) return baseValue;

if (memo.containsKey(year)) return memo.get(year);

double result = forecastMemo(year - 1) \* (1 + annualGrowthRate);

memo.put(year, result);

return result;

}

public static void main(String[] args) {

int forecastYears = 10;

System.out.println("Recursive Forecast (No Memoization):");

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

System.out.printf("Year %d: ₹%.2f%n", i, forecast(i));

}

System.out.println("\nRecursive Forecast (With Memoization):");

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

System.out.printf("Year %d: ₹%.2f%n", i, forecastMemo(i));

}

}

}

Output:

