Algorithms\_Data structures

Exercise 1: Inventory Management System

## Code:

### Product.java

package InventoryManagementSystem; public class Product {

private int productId;

private String productName; private int quantity;

private 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 int getProductId() { return productId; }

public String getProductName() { return productName; } public int getQuantity() { return quantity; }

public double getPrice() { return price; }

public void setProductName(String productName) { this.productName = productName; } public void setQuantity(int quantity) { this.quantity = quantity; }

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

public String toString() {

return "Product ID: " + productId + ", Name: " + productName +

", Quantity: " + quantity + ", Price: ₹" + price;

### InventoryManager.java

package InventoryManagementSystem; import java.util.HashMap;

public class InventoryManager {

private HashMap<Integer, Product> inventory; public InventoryManager() {

inventory = new HashMap<>();

}

public void addProduct(Product product) {

inventory.put(product.getProductId(), product); System.out.println("Product added successfully.");

}

public void updateProduct(int productId, String name, int quantity, double price) { Product product = inventory.get(productId);

if (product != null) {

product.setProductName(name); product.setQuantity(quantity);

product.setPrice(price);

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

} else {

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

}

}

public void deleteProduct(int productId) { if (inventory.remove(productId) != null) {

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

} else {

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

}

}

public void displayInventory() { if (inventory.isEmpty()) {

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

} else {

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

for (Product product : inventory.values()) { System.out.println(product);

}

}

}

}

### Main.java

package InventoryManagementSystem; import java.util.Scanner;

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager(); Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\n--- Inventory Management ---"); System.out.println("1. Add Product");

System.out.println("2. Update Product");

System.out.println("3. Delete Product"); System.out.println("4. Display Inventory"); System.out.println("5. Exit");

System.out.print("Choose an option: "); int choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter Product ID: "); int id = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter Product Name: "); String name = scanner.nextLine(); System.out.print("Enter Quantity: ");

int qty = scanner.nextInt(); System.out.print("Enter Price: ");

double price = scanner.nextDouble();

Product p = new Product(id, name, qty, price); manager.addProduct(p);

break;

case 2:

System.out.print("Enter Product ID to update: "); int uid = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter New Name: "); String uname = scanner.nextLine(); System.out.print("Enter New Quantity: "); int uqty = scanner.nextInt();

System.out.print("Enter New Price: "); double uprice = scanner.nextDouble();

manager.updateProduct(uid, uname, uqty, uprice); break;

case 3:

System.out.print("Enter Product ID to delete: "); int did = scanner.nextInt();

manager.deleteProduct(did); break;

case 4:

manager.displayInventory(); break;

case 5:

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

scanner.close(); System.exit(0); default:

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

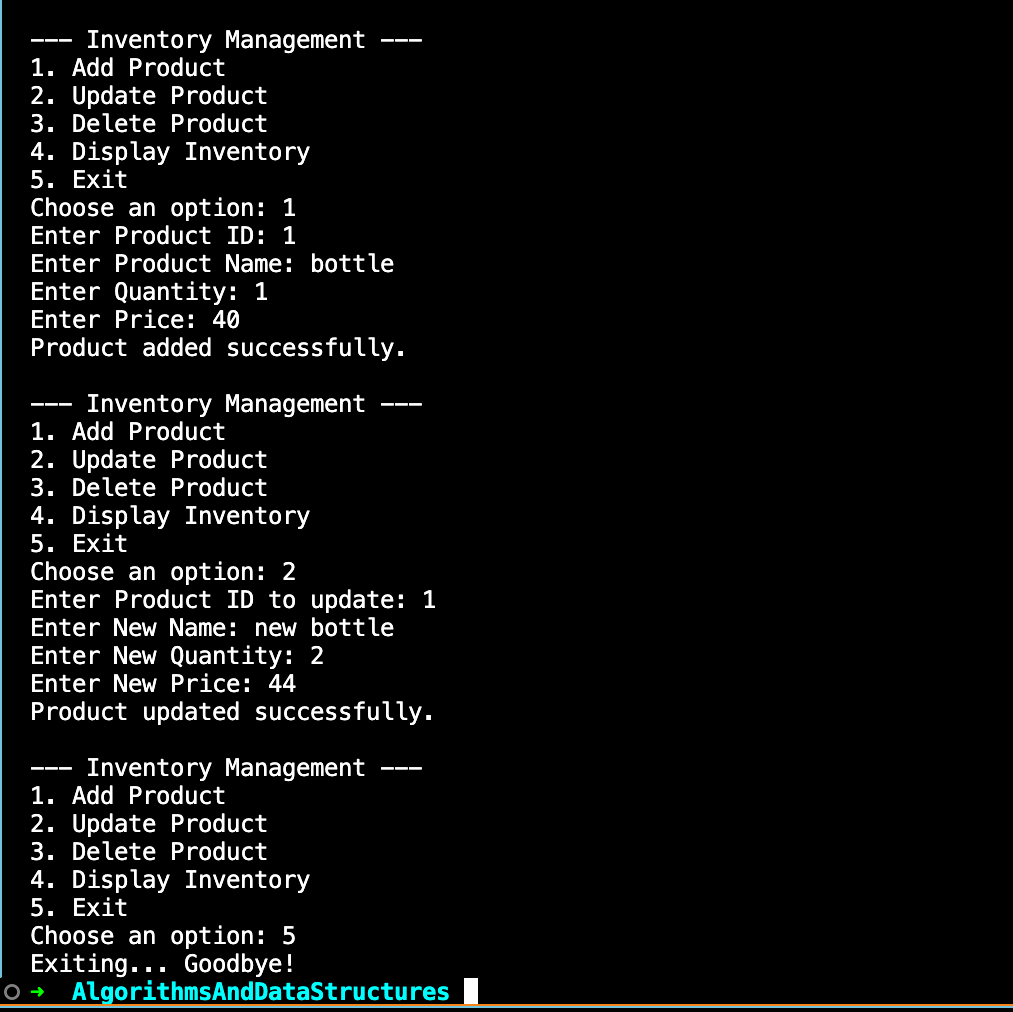
}

}

}

}

# output



Exercise 2: E-commerce Platform Search Function

## Code:

### Product.java

package Ecommerce; public class Product { private int productId;

private String productName; private String category;

public Product(int productId, String productName, String category) { this.productId = productId;

this.productName = productName; this.category = category;

}

public int getProductId() { return productId; }

public String getProductName() { return productName; } public String getCategory() { return category; } @Override

public String toString() {

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

}

}

### EcommerceSearch.java

package Ecommerce; import java.util.Arrays; import java.util.Comparator;

public class EcommerceSearch { public static void main(String[] args) { Product[] products = {

new Product(101, "Laptop", "Electronics"), new Product(58, "Sneakers", "Footwear"),

new Product(200, "Coffee Maker", "Home Appliance"), new Product(150, "Smartphone", "Electronics"),

new Product(75, "Backpack", "Accessories")

};

int targetId = 150;

// Linear search on unsorted array

int linearIndex = linearSearch(products, targetId); System.out.println("Linear Search: " +

(linearIndex != -1 ? products[linearIndex] : "Product not found"));

// Sort products by productId for binary search

Arrays.sort(products, Comparator.comparingInt(Product::getProductId));

// Binary search on sorted array

int binaryIndex = binarySearch(products, targetId); System.out.println("Binary Search: " +

(binaryIndex != -1 ? products[binaryIndex] : "Product not found"));

}

public static int linearSearch(Product[] products, int targetId) {

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

if (products[i].getProductId() == targetId) { return i;

}

}

return -1;

}

public static int binarySearch(Product[] products, int targetId) { int left = 0, right = products.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (products[mid].getProductId() == targetId) { return mid;

} else if (products[mid].getProductId() < targetId) {

left = mid + 1;

} else {

right = mid - 1;

}

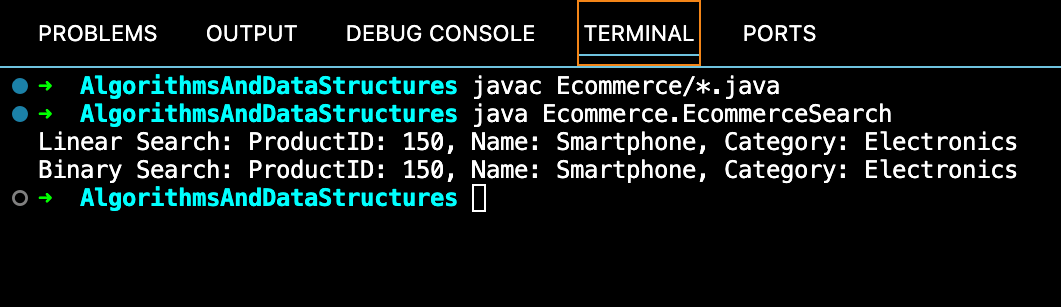
}

return -1;

}

}

# Output



Exercise 3: Sorting Customer Orders

## Code:

### Order.java

package SortingCustomerOrder; public class Order {

private int orderId;

private String customerName; private double totalPrice;

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

this.customerName = customerName; this.totalPrice = totalPrice;

}

public int getOrderId() { return orderId;

}

public String getCustomerName() { return customerName;

}

public double getTotalPrice() { return totalPrice;

}

@Override

public String toString() {

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

}

}

### SortingOrders.java

package SortingCustomerOrder; public class SortingOrders {

public 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].getTotalPrice() > orders[j+1].getTotalPrice()) { Order temp = orders[j];

orders[j] = orders[j+1]; orders[j+1] = temp;

}

}

}

}

// Quick Sort helper

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);

}

}

private static int partition(Order[] orders, int low, int high) { double pivot = orders[high].getTotalPrice();

int i = low - 1;

for (int j = low; j < high; j++) {

if (orders[j].getTotalPrice() <= 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 printOrders(Order[] orders) { for (Order order : orders) {

System.out.println(order);

}

}

public static void main(String[] args) { Order[] orders = {

new Order(101, "Alice", 2500.50), new Order(102, "Bob", 1500.00), new Order(103, "Charlie", 3000.75), new Order(104, "David", 1200.00), new Order(105, "Eve", 2800.00)

};

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

// Bubble Sort

bubbleSort(orders);

System.out.println("\nOrders after Bubble Sort:"); printOrders(orders);

// Quick Sort

// Shuffle back to original to show quicksort effect orders = new Order[]{

new Order(101, "Alice", 2500.50), new Order(102, "Bob", 1500.00), new Order(103, "Charlie", 3000.75), new Order(104, "David", 1200.00), new Order(105, "Eve", 2800.00)

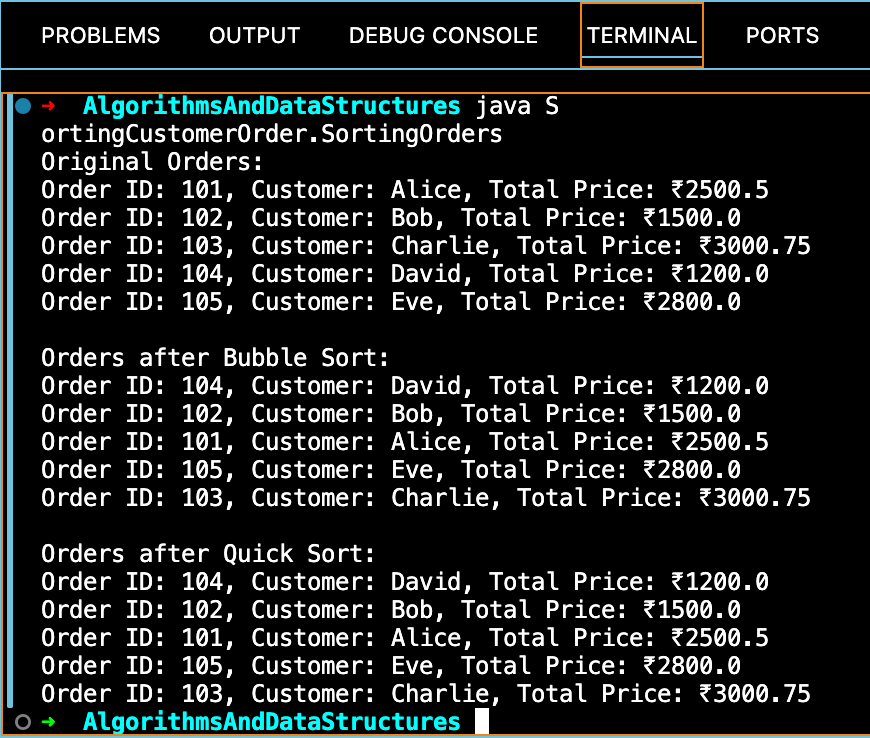
};

quickSort(orders, 0, orders.length - 1); System.out.println("\nOrders after Quick Sort:"); printOrders(orders);

}

}

# Output



Exercise 4: Employee Management System

## Code:

Employee.java

package EmployeeManagementSystem; public class Employee {

private int employeeId;

private String name; private String position; private 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 int getEmployeeId() { return employeeId;

}

public String getName() { return name;

}

public String getPosition() { return position;

}

public double getSalary() {

return salary;

}

@Override

public String toString() {

return "ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: ₹" + salary;

}

}

EmployeeManagementSystem.java

package EmployeeManagementSystem; import java.util.Scanner;

public class EmployeeManagementSystem { private Employee[] employees;

private int count;

public EmployeeManagementSystem(int size) { employees = new Employee[size];

count = 0;

}

public void addEmployee(Employee employee) { if (count < employees.length) {

employees[count++] = employee;

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

} else {

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

}

}

public Employee searchEmployee(int id) { for (int i = 0; i < count; i++) {

if (employees[i].getEmployeeId() == id) { return employees[i];

}

}

return null;

}

public void deleteEmployee(int id) { for (int i = 0; i < count; i++) {

if (employees[i].getEmployeeId() == id) { for (int j = i; j < count - 1; j++) {

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

}

employees[--count] = null;

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

}

}

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

}

public void traverseEmployees() { if (count == 0) {

System.out.println("No employees."); return;

}

System.out.println("Employee List:"); for (int i = 0; i < count; i++) { System.out.println(employees[i]);

}

}

public static void main(String[] args) { Scanner sc = new Scanner(System.in);

EmployeeManagementSystem system = new EmployeeManagementSystem(100);

while (true) {

System.out.println("\n--- Employee Management System ---"); System.out.println("1. Add Employee");

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

System.out.println("3. Delete Employee"); System.out.println("4. Show All Employees"); System.out.println("5. Exit");

System.out.print("Enter option: "); int option = sc.nextInt();

switch (option) { case 1:

System.out.print("Enter ID: "); int id = sc.nextInt();

sc.nextLine(); // consume newline

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();

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

case 2:

System.out.print("Enter ID to search: "); int sid = sc.nextInt();

Employee e = system.searchEmployee(sid);

if (e != null) {

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

} else {

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

}

break; case 3:

System.out.print("Enter ID to delete: "); int did = sc.nextInt(); system.deleteEmployee(did);

break; case 4:

system.traverseEmployees(); break;

case 5:

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

return; default:

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

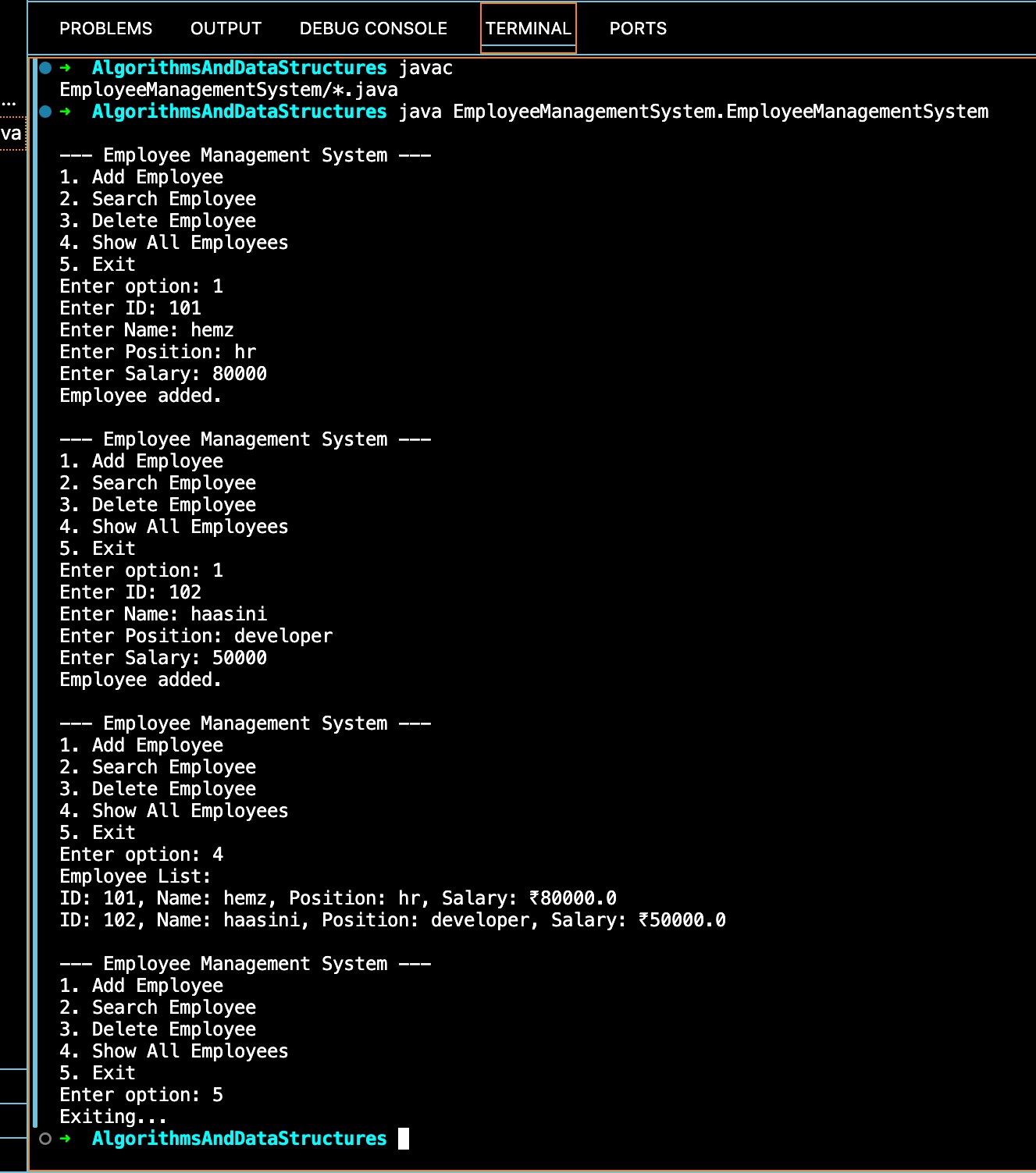
}

}

}

}

# output



Exercise 5: Task Management System

## Code:

### Task.java

package TaskManagementSystem; public class Task {

private int taskId; private String taskName; private String status;

public Task(int taskId, String taskName, String status) { this.taskId = taskId;

this.taskName = taskName; this.status = status;

}

public int getTaskId() { return taskId;

}

public String getTaskName() { return taskName;

}

public String getStatus() { return status;

}

@Override

public String toString() {

return "TaskID: " + taskId + ", Name: " + taskName + ", Status: " + status;

}

}

### TaskManagementSystem.java

package TaskManagementSystem; import java.util.Scanner;

class Node {

Task task;

Node next;

public Node(Task task) { this.task = task;

this.next = null;

}

}

public class TaskManagementSystem { private Node head;

public void addTask(Task task) {

Node newNode = new Node(task); if (head == null) {

head = newNode;

} else {

Node temp = head; while (temp.next != null) temp = temp.next;

temp.next = newNode;

}

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

}

public Task searchTask(int taskId) {

Node current = head; while (current != null) {

if (current.task.getTaskId() == taskId) return current.task;

current = current.next;

}

return null;

}

public void deleteTask(int taskId) { if (head == null) {

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

return;

}

if (head.task.getTaskId() == taskId) { head = head.next;

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

}

Node prev = head; Node curr = head.next; while (curr != null) {

if (curr.task.getTaskId() == taskId) { prev.next = curr.next;

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

return;

}

prev = curr;

curr = curr.next;

}

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

}

public void traverseTasks() { if (head == null) {

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

}

System.out.println("All Tasks:"); Node current = head;

while (current != null) { System.out.println(current.task); current = current.next;

}

}

public static void main(String[] args) { Scanner sc = new Scanner(System.in);

TaskManagementSystem system = new TaskManagementSystem(); while (true) {

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

System.out.println("1. Add Task"); System.out.println("2. Search Task"); System.out.println("3. Delete Task"); System.out.println("4. Show All Tasks");

System.out.println("5. Exit"); System.out.print("Enter choice: "); int choice = sc.nextInt();

switch (choice) { case 1:

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

int id = sc.nextInt(); sc.nextLine();

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

String name = sc.nextLine(); System.out.print("Enter Status: "); String status = sc.nextLine();

system.addTask(new Task(id, name, status)); break;

case 2:

System.out.print("Enter Task ID to search: "); int sid = sc.nextInt();

Task t = system.searchTask(sid); if (t != null)

System.out.println("Found: " + t); else

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

break; case 3:

System.out.print("Enter Task ID to delete: "); int did = sc.nextInt();

system.deleteTask(did); break;

case 4:

system.traverseTasks(); break;

case 5:

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

return;

default:

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

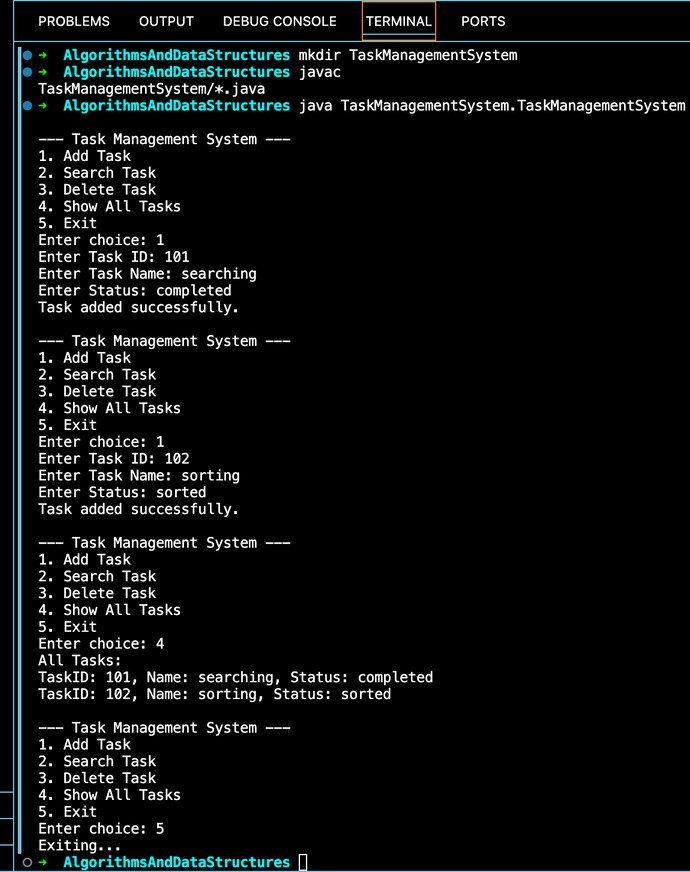
}

}

}

}

# output



Exercise 6: Library Management System

## Code:

### Book.java

package LibraryManagementSystem; public class Book {

private int bookId; private String title; private String author;

public Book(int bookId, String title, String author) { this.bookId = bookId;

this.title = title.toLowerCase(); // for case-insensitive comparison this.author = author;

}

public int getBookId() { return bookId;

}

public String getTitle() { return title;

}

public String getAuthor() { return author;

}

@Override

public String toString() {

return "BookID: " + bookId + ", Title: " + title + ", Author: " + author;

}

}

### LibraryManagementSystem.java

package LibraryManagementSystem; import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class LibraryManagementSystem {

public static Book linearSearch(Book[] books, String title) { for (Book book : books) {

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

return book;

}

}

return null;

}

public static Book binarySearch(Book[] books, String title) { int left = 0, right = books.length - 1;

title = title.toLowerCase();

while (left <= right) {

int mid = left + (right - left) / 2;

int cmp = books[mid].getTitle().compareTo(title); if (cmp == 0)

return books[mid];

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

right = mid - 1;

}

return null;

}

public static void main(String[] args) { Book[] books = {

new Book(101, "The Great Gatsby", "F. Scott Fitzgerald"),

new Book(102, "To Kill a Mockingbird", "Harper Lee"), new Book(103, "1984", "George Orwell"),

new Book(104, "Moby Dick", "Herman Melville"),

new Book(105, "Pride and Prejudice", "Jane Austen")

};

Scanner sc = new Scanner(System.in); System.out.println("\n--- Linear Search ---"); System.out.print("Enter book title to search: "); String title1 = sc.nextLine();

Book result1 = linearSearch(books, title1); if (result1 != null)

System.out.println("Found: " + result1); else

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

// Sort array before binary search

Arrays.sort(books, Comparator.comparing(Book::getTitle)); System.out.println("\n--- Binary Search ---");

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

String title2 = sc.nextLine();

Book result2 = binarySearch(books, title2);

if (result2 != null)

System.out.println("Found: " + result2); else

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

sc.close();

}

}

# output

Exercise 7: Financial Forecasting

## Code:

### FinancialForecast.java

package FinancialForecasting; public class FinancialForecast {

public static double futureValue(double presentValue, double rate, int years) { if (years == 0) {

return presentValue;

}

return futureValue(presentValue, rate, years - 1) \* (1 + rate);

}

// Optimized version using memoization

public static double futureValueMemo(double presentValue, double rate, int years, double[] memo) {

if (years == 0) return presentValue;

if (memo[years] != 0) return memo[years];

memo[years] = futureValueMemo(presentValue, rate, years - 1, memo) \* (1 + rate); return memo[years];

}

public static void main(String[] args) { double presentValue = 10000; // ₹10,000

double annualGrowthRate = 0.08; // 8% per year int forecastYears = 5;

System.out.println("=== Financial Forecast ==="); System.out.println("Present Value: ₹" + presentValue);

System.out.println("Annual Growth Rate: " + (annualGrowthRate \* 100) + "%");

// Recursive approach

double resultRecursive = futureValue(presentValue, annualGrowthRate, forecastYears);

System.out.printf("Future Value (Recursive) after %d years: ₹%.2f\n", forecastYears, resultRecursive);

// Optimized recursive with memoization

double[] memo = new double[forecastYears + 1];

double resultMemo = futureValueMemo(presentValue, annualGrowthRate, forecastYears, memo);

System.out.printf("Future Value (Memoized) after %d years: ₹%.2f\n", forecastYears, resultMemo);

}

}

output

