1

package *package\_assignment*;

*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;

}

@Override

*public* String toString() {

return employeeId + ": " + name + " - " + position + " (Rs. " + salary + ")";

}

}

*class* EmployeeManagement {

*static* Employee[] employees = new Employee[10];

*static* int count = 0;

*public* *static* void addEmployee(Employee emp) {

if (count < employees.length) {

employees[count++] = emp;

} else {

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

}

}

*public* *static* Employee searchEmployee(int id) {

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

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

return employees[i];

}

}

return null;

}

*public* *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];

}

employees[--count] = null;

break;

}

}

}

*public* *static* void displayEmployees() {

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

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

}

}

*public* *static* void main(String[] args) {

addEmployee(new Employee(1, "Alice", "Manager", 50000));

addEmployee(new Employee(2, "Bob", "Developer", 35000));

addEmployee(new Employee(3, "Charlie", "Designer", 30000));

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

displayEmployees();

System.out.println("\n--- Searching for Employee with ID 2 ---");

Employee found = searchEmployee(2);

System.out.println(found != null ? found : "Not Found");

System.out.println("\n--- Deleting Employee with ID 1 ---");

deleteEmployee(1);

displayEmployees();

}

}

OUTPUT:

--- Employee List ---

1: Alice - Manager (Rs. 50000.0)

2: Bob - Developer (Rs. 35000.0)

3: Charlie - Designer (Rs. 30000.0)

--- Searching for Employee with ID 2 ---

2: Bob - Developer (Rs. 35000.0)

--- Deleting Employee with ID 1 ---

2: Bob - Developer (Rs. 35000.0)

3: Charlie - Designer (Rs. 30000.0)

2

package *package\_assignment*;

*class* Forecasting {

*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* double predictMemoized(double currentValue, double growthRate, int years, Double[] memo) {

if (years == 0) return currentValue;

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

memo[years] = predictMemoized(currentValue \* (1 + growthRate), growthRate, years - 1, memo);

return memo[years];

}

*public* *static* void main(String[] args) {

double initialValue = 10000;

double growthRate = 0.08;

int years = 5;

double result = predictValue(initialValue, growthRate, years);

System.out.println("Recursive Forecast: Rs. " + result);

Double[] memo = new Double[years + 1];

double memoResult = predictMemoized(initialValue, growthRate, years, memo);

System.out.println("Optimized Forecast (Memoized): Rs. " + memoResult);

}

}

Recursive Forecast: Rs. 14693.282399999999

Optimized Forecast (Memoized): Rs. 14693.282399999999

3

package *package\_assignment*;

import *java.util.HashMap*;

import *java.util.Map*;

*class* Product {

String productId;

String productName;

int quantity;

double price;

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

@Override

*public* String toString() {

return productName + " [ID: " + productId + ", Qty: " + quantity + ", Price: Rs. " + price + "]";

}

}

*public* *class* InventoryApp {

*static* Map<String, Product> inventory = new HashMap<>();

*public* *static* void addProduct(Product p) {

inventory.put(p.productId, p);

}

*public* *static* void updateProduct(String id, int newQty, double newPrice) {

Product p = inventory.get(id);

if (p != null) {

p.quantity = newQty;

p.price = newPrice;

}

}

*public* *static* void deleteProduct(String id) {

inventory.remove(id);

}

*public* *static* void displayInventory() {

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

System.out.println(p);

}

}

*public* *static* void main(String[] args) {

addProduct(new Product("P01", "Keyboard", 10, 999.99));

addProduct(new Product("P02", "Mouse", 25, 499.50));

addProduct(new Product("P03", "Monitor", 5, 7500.00));

System.out.println("--- Current Inventory ---");

displayInventory();

updateProduct("P02", 30, 459.99);

System.out.println("\n--- After Update ---");

displayInventory();

deleteProduct("P01");

System.out.println("\n--- After Deletion ---");

displayInventory();

}

}

--- Current Inventory ---

Keyboard [ID: P01, Qty: 10, Price: Rs. 999.99]

Mouse [ID: P02, Qty: 25, Price: Rs. 499.5]

Monitor [ID: P03, Qty: 5, Price: Rs. 7500.0]

--- After Update ---

Keyboard [ID: P01, Qty: 10, Price: Rs. 999.99]

Mouse [ID: P02, Qty: 30, Price: Rs. 459.99]

Monitor [ID: P03, Qty: 5, Price: Rs. 7500.0]

--- After Deletion ---

Mouse [ID: P02, Qty: 30, Price: Rs. 459.99]

Monitor [ID: P03, Qty: 5, Price: Rs. 7500.0]

4

package *package\_assignment*;

import *java.util.Arrays*;

*class* Book {

String bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

*public* String toString() {

return title + " by " + author + " (ID: " + bookId + ")";

}

}

*class* Library {

*static* Book[] books = {

new Book("B101", "Java Basics", "James Gosling"),

new Book("B102", "Effective Java", "Joshua Bloch"),

new Book("B103", "Clean Code", "Robert C. Martin"),

new Book("B104", "Python Tricks", "Dan Bader"),

new Book("B105", "Data Structures", "Mark Allen Weiss")

};

*public* *static* Book linearSearch(String title) {

for (Book b : books) {

if (b.title.equalsIgnoreCase(title)) {

return b;

}

}

return null;

}

*public* *static* Book binarySearch(String title) {

Arrays.sort(books, (a, b) -> a.title.compareToIgnoreCase(b.title));

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

while (left <= right) {

int mid = (left + right) / 2;

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

if (cmp == 0) return books[mid];

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

else right = mid - 1;

}

return null;

}

*public* *static* void main(String[] args) {

String searchTitle = "Clean Code";

Book linear = linearSearch(searchTitle);

System.out.println("Linear Search: " + (linear != null ? linear : "Not Found"));

Book binary = binarySearch(searchTitle);

System.out.println("Binary Search: " + (binary != null ? binary : "Not Found"));

}

}

Linear Search: Clean Code by Robert C. Martin (ID: B103)

Binary Search: Clean Code by Robert C. Martin (ID: B103)

5

package *package\_assignment*;

import *java.util.Arrays*;

import *java.util.Scanner*;

*class* ProductSearch {

String productId;

String productName;

String category;

ProductSearch(String productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

@Override

*public* String toString() {

return productName + " (" + category + ") - ID: " + productId;

}

}

*class* SearchDemo {

*static* ProductSearch[] products = {

new ProductSearch("101", "Laptop", "Electronics"),

new ProductSearch("102", "Smartphone", "Electronics"),

new ProductSearch("103", "Shoes", "Fashion"),

new ProductSearch("104", "Watch", "Accessories"),

new ProductSearch("105", "Keyboard", "Electronics")

};

*public* *static* ProductSearch linearSearch(String name) {

for (ProductSearch p : products) {

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

return p;

}

}

return null;

}

*public* *static* ProductSearch binarySearch(String name) {

Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

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

while (left <= right) {

int mid = (left + right) / 2;

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

if (cmp == 0) return products[mid];

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

else right = mid - 1;

}

return null;

}

*public* *static* void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String name = sc.nextLine();

ProductSearch result1 = linearSearch(name);

System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Not Found"));

ProductSearch result2 = binarySearch(name);

System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Not Found"));

sc.close();

}

}

Enter product name to search: Table

Linear Search Result: Not Found

Binary Search Result: Not Found

6

package *package\_assignment*;

*class* Order {

String orderId;

String customerName;

double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

@Override

*public* String toString() {

return "OrderID: " + orderId + ", Customer: " + customerName + ", Total: Rs. " + totalPrice;

}

}

*class* SortingOrders {

*public* *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;

}

}

}

}

*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 printOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

}

*public* *static* void main(String[] args) {

Order[] orders = {

new Order("O01", "Alice", 1200.00),

new Order("O02", "Bob", 4500.50),

new Order("O03", "Charlie", 2999.99),

new Order("O04", "David", 800.00)

};

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

printOrders(orders);

System.out.println("\n--- Orders Sorted by Bubble Sort ---");

bubbleSort(orders);

printOrders(orders);

Order[] quickSorted = {

new Order("O01", "Alice", 1200.00),

new Order("O02", "Bob", 4500.50),

new Order("O03", "Charlie", 2999.99),

new Order("O04", "David", 800.00)

};

System.out.println("\n--- Orders Sorted by Quick Sort ---");

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

printOrders(quickSorted);

}

}

--- Original Orders ---

OrderID: O01, Customer: Alice, Total: Rs. 1200.0

OrderID: O02, Customer: Bob, Total: Rs. 4500.5

OrderID: O03, Customer: Charlie, Total: Rs. 2999.99

OrderID: O04, Customer: David, Total: Rs. 800.0

--- Orders Sorted by Bubble Sort ---

OrderID: O04, Customer: David, Total: Rs. 800.0

OrderID: O01, Customer: Alice, Total: Rs. 1200.0

OrderID: O03, Customer: Charlie, Total: Rs. 2999.99

OrderID: O02, Customer: Bob, Total: Rs. 4500.5

--- Orders Sorted by Quick Sort ---

OrderID: O04, Customer: David, Total: Rs. 800.0

OrderID: O01, Customer: Alice, Total: Rs. 1200.0

OrderID: O03, Customer: Charlie, Total: Rs. 2999.99

OrderID: O02, Customer: Bob, Total: Rs. 4500.5

7

package *package\_assignment*;

*class* Task {

int taskId;

String taskName;

String status;

Task next;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

@Override

*public* String toString() {

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

}

}

*class* TaskLinkedList {

Task head = null;

*public* void addTask(Task task) {

if (head == null) {

head = task;

} else {

Task temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = task;

}

}

*public* Task searchTask(int taskId) {

Task temp = head;

while (temp != null) {

if (temp.taskId == taskId) return temp;

temp = temp.next;

}

return null;

}

*public* void deleteTask(int taskId) {

if (head == null) return;

if (head.taskId == taskId) {

head = head.next;

return;

}

Task temp = head;

while (temp.next != null) {

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

temp.next = temp.next.next;

return;

}

temp = temp.next;

}

}

*public* void traverseTasks() {

Task temp = head;

while (temp != null) {

System.out.println(temp);

temp = temp.next;

}

}

*public* *static* void main(String[] args) {

TaskLinkedList taskList = new TaskLinkedList();

taskList.addTask(new Task(1, "Design Module", "Pending"));

taskList.addTask(new Task(2, "Implement Feature", "In Progress"));

taskList.addTask(new Task(3, "Testing", "Completed"));

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

taskList.traverseTasks();

System.out.println("\n--- Searching for Task ID 2 ---");

Task found = taskList.searchTask(2);

System.out.println(found != null ? found : "Not Found");

System.out.println("\n--- Deleting Task ID 1 ---");

taskList.deleteTask(1);

taskList.traverseTasks();

}

}

--- Task List ---

TaskID: 1, Name: Design Module, Status: Pending

TaskID: 2, Name: Implement Feature, Status: In Progress

TaskID: 3, Name: Testing, Status: Completed

--- Searching for Task ID 2 ---

TaskID: 2, Name: Implement Feature, Status: In Progress

--- Deleting Task ID 1 ---

TaskID: 2, Name: Implement Feature, Status: In Progress

TaskID: 3, Name: Testing, Status: Completed