**DINIYA**

**Exercise 1: Inventory Management System**

CODE:

public class InventoryManagementSystem {

public static void main(String[] args) {

Inventory inventory = new Inventory();

inventory.addProduct(1, "Product A", 10, 100.0);

inventory.addProduct(2, "Product B", 20, 200.0);

inventory.updateProduct(1, "Product A1", 15, 150.0);

inventory.deleteProduct(2);

inventory.displayProducts();

}

}

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 void setProductId(int productId) {

this.productId = productId;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

@Override

public String toString() {

return "Product{" +

"productId=" + productId +

", productName='" + productName + '\'' +

", quantity=" + quantity +

", price=" + price +

'}';

}

}

**Inventory Class:**

import java.util.HashMap;

import java.util.Map;

class Inventory {

private Map<Integer, Product> products;

public Inventory() {

products = new HashMap<>();

}

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

Product product = new Product(productId, productName, quantity, price);

products.put(productId, product);

System.out.println("Product added: " + product);

}

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

Product product = products.get(productId);

if (product != null) {

product.setProductName(productName);

product.setQuantity(quantity);

product.setPrice(price);

System.out.println("Product updated: " + product);

} else {

System.out.println("Product not found with ID: " + productId);

}

}

public void deleteProduct(int productId) {

Product removedProduct = products.remove(productId);

if (removedProduct != null) {

System.out.println("Product removed: " + removedProduct);

} else {

System.out.println("Product not found with ID: " + productId);

}

}

public void displayProducts() {

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

System.out.println(product);

}

}

}

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

CODE:

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 void setProductId(int productId) {

this.productId = productId;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

}

public class Search {

public static Product linearSearch(Product[] products, String productName) {

for (Product product : products) {

if (product.getProductName().equalsIgnoreCase(productName)) {

return product;

}

}

return null;

}

}

USING BINARY SEARCH:

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

public static Product binarySearch(Product[] products, String productName) {

Arrays.sort(products, Comparator.comparing(Product::getProductName)); // Ensure the array is sorted

int left = 0;

int right = products.length - 1;

while (left <= right) {

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

int comparison = products[middle].getProductName().compareToIgnoreCase(productName);

if (comparison == 0) {

return products[middle];

} else if (comparison < 0) {

left = middle + 1;

} else {

right = middle - 1;

}

}

return null;

}

}

**Exercise 3: Sorting Customer Orders**

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 void setOrderId(int orderId) {

this.orderId = orderId;

}

public String getCustomerName() {

return customerName;

}

public void setCustomerName(String customerName) {

this.customerName = customerName;

}

public double getTotalPrice() {

return totalPrice;

}

public void setTotalPrice(double totalPrice) {

this.totalPrice = totalPrice;

}

}

public class BubbleSort {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

boolean swapped;

for (int i = 0; i < n - 1; i++) {

swapped = false;

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;

swapped = true;

}

}

if (!swapped) break;

}

}

}

public class QuickSort {

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); // Index of smaller element

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;

}

}

**Exercise 4: Employee Management System**

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 void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

public class EmployeeManagement {

private Employee[] employees;

private int size;

public EmployeeManagement(int capacity) {

employees = new Employee[capacity];

size = 0;

}

public void addEmployee(Employee employee) {

if (size >= employees.length) {

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

return;

}

employees[size++] = employee;

}

public Employee searchEmployee(int employeeId) {

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

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

return employees[i];

}

}

return null; // Employee not found

}

public void traverseEmployees() {

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

System.out.println("ID: " + employees[i].getEmployeeId() + ", Name: " + employees[i].getName() + ", Position: " + employees[i].getPosition() + ", Salary: " + employees[i].getSalary());

}

}

public void deleteEmployee(int employeeId) {

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

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

employees[i] = employees[size - 1]; // Replace with last element

employees[size - 1] = null;

size--;

return;

}

}

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

}

}

**Exercise 5: Task Management System**

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 void setTaskId(int taskId) {

this.taskId = taskId;

}

public String getTaskName() {

return taskName;

}

public void setTaskName(String taskName) {

this.taskName = taskName;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

class Node {

Task task;

Node next;

public Node(Task task) {

this.task = task;

this.next = null;

}

}

public class TaskManagement {

private Node head;

public TaskManagement() {

this.head = null;

}

public void addTask(Task task) {

Node newNode = new Node(task);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public Task searchTask(int taskId) {

Node current = head;

while (current != null) {

if (current.task.getTaskId() == taskId) {

return current.task;

}

current = current.next;

}

return null; // Task not found

}

public void traverseTasks() {

Node current = head;

while (current != null) {

System.out.println("ID: " + current.task.getTaskId() + ", Name: " + current.task.getTaskName() + ", Status: " + current.task.getStatus());

current = current.next;

}

}

public void deleteTask(int taskId) {

if (head == null) {

return;

}

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

head = head.next;

return;

}

Node current = head;

while (current.next != null && current.next.task.getTaskId() != taskId) {

current = current.next;

}

if (current.next != null) {

current.next = current.next.next;

}

}

}

**Exercise 6: Library Management System**

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;

this.author = author;

}

public int getBookId() {

return bookId;

}

public void setBookId(int bookId) {

this.bookId = bookId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

}

public class LinearSearch {

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

for (Book book : books) {

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

return book;

}

}

return null; // Book not found

}

}

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

public static Book binarySearch(Book[] books, String title) {

Arrays.sort(books, Comparator.comparing(Book::getTitle)); // Ensure the array is sorted

int left = 0;

int right = books.length - 1;

while (left <= right) {

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

int comparison = books[middle].getTitle().compareToIgnoreCase(title);

if (comparison == 0) {

return books[middle];

} else if (comparison < 0) {

left = middle + 1;

} else {

right = middle - 1;

}

}

return null;

}

}

**Exercise 7: Financial Forecasting**

public class FinancialForecasting {

public static double calculateFutureValue(double presentValue, double growthRate, int periods) {

if (periods == 0) {

return presentValue;

}

return calculateFutureValue(presentValue \* (1 + growthRate), growthRate, periods - 1);

}

}

public class FinancialForecasting {

public static double calculateFutureValue(double presentValue, double growthRate, int periods) {

if (periods == 0) {

return presentValue;

}

return calculateFutureValue(presentValue \* (1 + growthRate), growthRate, periods - 1);

}

public static void main(String[] args) {

double presentValue = 1000.0;

double growthRate = 0.05; // 5%

int periods = 10;

double futureValue = calculateFutureValue(presentValue, growthRate, periods);

System.out.println("Future Value: " + futureValue);

}

}

public class FinancialForecasting {

public static double calculateFutureValue(double presentValue, double growthRate, int periods) {

double futureValue = presentValue;

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

futureValue \*= (1 + growthRate);

}

return futureValue;

}

public static void main(String[] args) {

double presentValue = 1000.0;

double

Top of Form

Bottom of Form