**CHIDURA SREENIDHI**

**C#- ASSIGNMENT**

**COURIER MANAGEMENT SYSTEM**

using System;

namespace Console\_Hexaware

{

internal class Assignment

{

//Task 1: Control Flow Statements

//order delivery status

static void Main()

{

string Delivered = "Delivered";

string Cancelled = "Cancelled";

string Processing = "Processing";

Console.Write("Enter order status (Processing/Delivered/Cancelled): ");

string status = Console.ReadLine();

if (status.Equals("Delivered", StringComparison.OrdinalIgnoreCase))

Console.WriteLine("The order has been delivered.");

else if (status.Equals("Processing", StringComparison.OrdinalIgnoreCase))

Console.WriteLine("The order is being processed.");

else if (status.Equals("Cancelled", StringComparison.OrdinalIgnoreCase))

Console.WriteLine("The order was cancelled.");

else

Console.WriteLine("Invalid status.");

//parcel weight Category using switch case

Console.Write("Enter parcel weight in kg: ");

double weight = Convert.ToDouble(Console.ReadLine());

string category;

if (weight < 1)

{

category = "Light";

}

else if (weight >= 1 && weight <= 5)

{

category = "Medium";

}

else

{

category = "Heavy";

}

switch (category)

{

case "Light":

Console.WriteLine("Category: Light");

break;

case "Medium":

Console.WriteLine("Category: Medium");

break;

case "Heavy":

Console.WriteLine("Category: Heavy");

break;

}

//User Authentication

string empEmail = "emp@courier.com", empPass = "emp123";

string custEmail = "cust@courier.com", custPass = "cust123";

Console.Write("Enter Email: ");

string email = Console.ReadLine();

Console.Write("Enter Password: ");

string password = Console.ReadLine();

if (email == empEmail && password == empPass)

{

Console.WriteLine("Welcome, Employee!");

}

else if (email == custEmail && password == custPass)

{

Console.WriteLine("Welcome, Customer!");

}

else

{

Console.WriteLine("Invalid login!");

}

//assigning courirers to shipments

Console.Write("Enter number of couriers: ");

int n = int.Parse(Console.ReadLine());

string[] courierNames = new string[n];

int[] proximity = new int[n]; // Lower value = closer

int[] load = new int[n]; // Lower value = less busy

for (int i = 0; i < n; i++)

{

Console.WriteLine($"Enter details for courier {i + 1}:");

Console.Write("Name: ");

courierNames[i] = Console.ReadLine();

Console.Write("Proximity (e.g., distance in km): ");

proximity[i] = int.Parse(Console.ReadLine());

Console.Write("Load (no. of parcels currently assigned): ");

load[i] = int.Parse(Console.ReadLine());

}

int bestIndex = 0;

for (int i = 1; i < n; i++)

{

if (proximity[i] < proximity[bestIndex])

{

bestIndex = i;

}

else if (proximity[i] == proximity[bestIndex])

{

if (load[i] < load[bestIndex])

{

bestIndex = i;

}

}

}

Console.WriteLine($"Courier assigned: {courierNames[bestIndex]}");

Console.WriteLine($"Proximity: {proximity[bestIndex]} km, Load: {load[bestIndex]} parcels");

//Task 2: Loops and Iteration

//display orders of a customer

Console.Write("Enter customer name: ");

string customerName = Console.ReadLine();

Console.Write($"How many orders does {customerName} have? ");

int orderCount = int.Parse(Console.ReadLine());

string[] orders = new string[orderCount];

for (int i = 0; i < orderCount; i++)

{

Console.Write($"Enter order {i + 1} details: ");

orders[i] = Console.ReadLine();

}

Console.WriteLine($"All orders of customer: {customerName}");

for (int i = 0; i < orders.Length; i++)

{

Console.WriteLine($"Order {i + 1}: {orders[i]}");

}

//real - time Courier location

Console.Write("Enter Order Date (yyyy-MM-dd): ");

DateTime orderDate;

while (!DateTime.TryParse(Console.ReadLine(), out orderDate))

{

Console.Write("Invalid format. Enter date again (yyyy-MM-dd): ");

}

DateTime currentDate = DateTime.Today;

int daysPassed = (currentDate - orderDate).Days;

Console.WriteLine($"\n Days since order placed: {daysPassed}");

if (daysPassed <= 2)

Console.WriteLine(" Parcel Status: In Warehouse");

else if (daysPassed < 5)

Console.WriteLine("Parcel Status: In Transit");

else if (daysPassed < 7)

Console.WriteLine("Parcel Status: Near Destination");

else

Console.WriteLine("Parcel Status: Delivered");

//Task 3: Arrays and Data Structures

//Create an array to store the tracking history of a parcel

Console.Write("Enter number of location updates: ");

int n = int.Parse(Console.ReadLine());

string[] trackingHistory = new string[n];

for (int i = 0; i < n; i++)

{

Console.Write($"Enter location update #{i + 1}: ");

trackingHistory[i] = Console.ReadLine();

}

Console.WriteLine("Tracking History:");

for (int i = 0; i < trackingHistory.Length; i++)

{

Console.WriteLine($"Update {i + 1}: {trackingHistory[i]}");

}

//Implement a method to find the nearest available courier

Console.Write("Enter number of couriers: ");

int n = int.Parse(Console.ReadLine());

string[] names = new string[n];

int[] distances = new int[n];

bool[] available = new bool[n];

for (int i = 0; i < n; i++)

{

Console.WriteLine($"\n Courier {i + 1}:");

Console.Write("Name: ");

names[i] = Console.ReadLine();

Console.Write("Distance to pickup location (in km): ");

distances[i] = int.Parse(Console.ReadLine());

Console.Write("Is Available? (yes/no): ");

available[i] = Console.ReadLine().ToLower() == "yes";

}

int bestIndex = -1;

for (int i = 0; i < n; i++)

{

if (available[i])

{

if (bestIndex == -1 || distances[i] < distances[bestIndex])

{

bestIndex = i;

}

}

}

if (bestIndex != -1)

{

Console.WriteLine($"Nearest Available Courier: {names[bestIndex]} ({distances[bestIndex]} km away)");

}

else

{

Console.WriteLine(" No available couriers found.");

}

//Task 4: Strings,2d Arrays, user defined functions,Hashmap

// Parcel Tracking Using 2D Array

string[,] parcels = {

{ "TRK123", "In Transit" },

{ "TRK456", "Out for Delivery" },

{ "TRK789", "Delivered" }

};

Console.Write("Enter Tracking Number: ");

string input = Console.ReadLine();

bool found = false;

for (int i = 0; i < parcels.GetLength(0); i++)

{

if (parcels[i, 0].Equals(input, StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine($"📦 Status for {input}: {parcels[i, 1]}");

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Tracking number not found.");

}

//Customer Data Validation Function

using System;

using System.Text.RegularExpressions;

namespace Console\_Hexaware

{

class CustomerValidator

{

static bool Validate(string data, string detailType)

{

switch (detailType.ToLower())

{

case "name":

return Regex.IsMatch(data, @"^[A-Z][a-zA-Z\s]\*$");

case "address":

return Regex.IsMatch(data, @"^[a-zA-Z0-9\s,.-]+$");

case "phone":

return Regex.IsMatch(data, @"^\d{3}-\d{3}-\d{4}$");

default:

return false;

}

}

static void Main()

{

Console.Write("Enter data to validate: ");

string data = Console.ReadLine();

Console.Write("\nEnter detail type (name, address, phone): ");

string type = Console.ReadLine();

if (Validate(data, type))

Console.WriteLine("Valid " + type);

else

Console.WriteLine("Invalid " + type);

}

}

}

// Adress formatting

using System;

using System.Globalization;

class AddressFormatter

{

static string FormatAddress(string street, string city, string state, string zipCode)

{

// Capitalize first letter of each word

TextInfo textInfo = CultureInfo.CurrentCulture.TextInfo;

street = textInfo.ToTitleCase(street.ToLower());

city = textInfo.ToTitleCase(city.ToLower());

state = state.ToUpper(); // state as abbreviation

zipCode = zipCode.PadLeft(5, '0'); // ensure 5 digits

return $"{street}, {city}, {state} - {zipCode}";

}

static void Main()

{

Console.Write("Enter Street: ");

string street = Console.ReadLine();

Console.Write("Enter City: ");

string city = Console.ReadLine();

Console.Write("Enter State (e.g., CA): ");

string state = Console.ReadLine();

Console.Write("Enter Zip Code: ");

string zip = Console.ReadLine();

string formatted = FormatAddress(street, city, state, zip);

Console.WriteLine("\n📬 Formatted Address:");

Console.WriteLine(formatted);

}

}

//Order confirmation email

using System;

class OrderConfirmation

{

static string GenerateEmail(string name, string orderNo, string address, DateTime orderDate, DateTime deliveryDate)

{

return $"Hello {name},\n\n" +

$"Your order (#{orderNo}) has been placed successfully.\n" +

$"Delivery Address: {address}\n" +

$"📆 Ordered Date: {orderDate:dd MMM yyyy}\n" +

$"Expected Delivery Date: {deliveryDate:dd MMM yyyy}\n\n" +

"Thank you for using our courier service!";

}

static void Main()

{

Console.Write("Enter Customer Name: ");

string name = Console.ReadLine();

Console.Write("Enter Order Number: ");

string orderNo = Console.ReadLine();

Console.Write("Enter Delivery Address: ");

string address = Console.ReadLine();

Console.Write("Enter Order Date (yyyy-MM-dd): ");

DateTime orderDate = DateTime.Parse(Console.ReadLine()); // direct parsing

DateTime expectedDelivery = orderDate.AddDays(3);

string email = GenerateEmail(name, orderNo, address, orderDate, expectedDelivery);

Console.WriteLine("\n📧 Order Confirmation Email:\n");

Console.WriteLine(email);

}

}

//Calculate Shipping Costs

using System;

class ShippingCostCalculator

{

static double CalculateShippingCost(double distance, double weight)

{

double baseRate = 5.0;

double distanceRate = 2.0; // per km

double weightRate = 3.0; // per kg

return baseRate + (distance \* distanceRate) + (weight \* weightRate);

}

static void Main()

{

Console.Write("Enter distance (in km): ");

double distance = double.Parse(Console.ReadLine());

Console.Write("Enter parcel weight (in kg): ");

double weight = double.Parse(Console.ReadLine());

double cost = CalculateShippingCost(distance, weight);

Console.WriteLine($"🚚 Shipping Cost: ₹{cost:F2}");

}

}

//Password generator

using System;

class PasswordGenerator

{

static string GeneratePassword(int length)

{

string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

string lower = "abcdefghijklmnopqrstuvwxyz";

string digits = "0123456789";

string special = "!@#$%^&\*";

string all = upper + lower + digits + special;

Random rand = new Random();

char[] password = new char[length];

for (int i = 0; i < length; i++)

{

password[i] = all[rand.Next(all.Length)];

}

return new string(password);

}

static void Main()

{

Console.Write("Enter desired password length: ");

int length = int.Parse(Console.ReadLine());

string password = GeneratePassword(length);

Console.WriteLine($"🔐 Generated Password: {password}");

}

}

//Find Similar Addresses

using System;

class SimilarAddressFinder

{

static void Main()

{

string[] addresses = {

"123 Main Street",

"123 Main St.",

"456 Oak Avenue",

"123 main street",

"789 Maple Blvd"

};

Console.Write("Enter address to search: ");

string input = Console.ReadLine().ToLower();

bool found = false;

Console.WriteLine("\n Similar Addresses:");

foreach (string addr in addresses)

{

if (addr.ToLower().Contains(input))

{

Console.WriteLine(addr);

found = true;

}

}

if (!found)

{

Console.WriteLine(" No found ");

}

}

}

//Task 5: Object-Oriented Programming – Model/Entity Classes

//1.User.cs

using System;

public class User

{

private int userID;

private string userName, email, password, contactNumber, address;

public User() { }

public User(int userID, string userName, string email, string password, string contactNumber, string address)

{

this.userID = userID;

this.userName = userName;

this.email = email;

this.password = password;

this.contactNumber = contactNumber;

this.address = address;

}

public int UserID { get => userID; set => userID = value; }

public string UserName { get => userName; set => userName = value; }

public string Email { get => email; set => email = value; }

public string Password { get => password; set => password = value; }

public string ContactNumber { get => contactNumber; set => contactNumber = value; }

public string Address { get => address; set => address = value; }

public override string ToString()

{

return $"User: {userName}, Email: {email}, Contact: {contactNumber}, Address: {address}";

}

}

//2. Courier.cs

public class Courier

{

private static long trackingSeed = 100000;

private long courierID;

private string senderName, senderAddress, receiverName, receiverAddress, status;

private double weight;

private long trackingNumber;

private DateTime deliveryDate;

private int userId;

public Courier()

{

this.trackingNumber = ++trackingSeed;

}

public Courier(long courierID, string senderName, string senderAddress, string receiverName,

string receiverAddress, double weight, string status, DateTime deliveryDate, int userId)

{

this.courierID = courierID;

this.senderName = senderName;

this.senderAddress = senderAddress;

this.receiverName = receiverName;

this.receiverAddress = receiverAddress;

this.weight = weight;

this.status = status;

this.deliveryDate = deliveryDate;

this.userId = userId;

this.trackingNumber = ++trackingSeed;

}

public long TrackingNumber => trackingNumber;

public override string ToString()

{

return $"Courier {trackingNumber}: {senderName} ➡ {receiverName}, Status: {status}";

}

}

//3. Employee.cs

public class Employee

{

private int employeeID;

private string employeeName, email, contactNumber, role;

private double salary;

public Employee() { }

public Employee(int id, string name, string email, string contact, string role, double salary)

{

this.employeeID = id;

this.employeeName = name;

this.email = email;

this.contactNumber = contact;

this.role = role;

this.salary = salary;

}

}

//4. Location.cs

public class Location

{

private int locationID;

private string locationName, address;

public Location() { }

public Location(int id, string name, string address)

{

this.locationID = id;

this.locationName = name;

this.address = address;

}

}

//5. CourierCompany.cs

public class CourierCompany

{

public string companyName;

public Courier[] courierDetails;

public Employee[] employeeDetails;

public Location[] locationDetails;

}

//6. Payment.cs

public class Payment

{

public long PaymentID { get; set; }

public long CourierID { get; set; }

public double Amount { get; set; }

public DateTime PaymentDate { get; set; }

public override string ToString()

{

return $"Payment {PaymentID}: ₹{Amount} for Courier {CourierID} on {PaymentDate:dd MMM yyyy}";

}

}

class Program

{

static void Main()

{

Console.WriteLine("User Object:");

User user = new User(1, "Nidhi", "nidhi@email.com", "pass123", "9876543210", "Hyderabad");

Console.WriteLine(user);

Console.WriteLine("\nCourier Object:");

Courier courier = new Courier(101, "Ravi", "Hyderabad", "Neha", "Mumbai", 2.5, "Processing", DateTime.Now.AddDays(3), user.UserID);

Console.WriteLine(courier);

Console.WriteLine("\nEmployee Object:");

Employee employee = new Employee(201, "Karan", "karan@company.com", "9998887777", "Delivery Staff", 25000);

Console.WriteLine(employee);

Console.WriteLine("\nLocation Object:");

Location location = new Location(301, "Banjara Hills Hub", "Banjara Hills, Hyderabad");

Console.WriteLine(location);

Console.WriteLine("\nPayment Object:");

Payment payment = new Payment

{

PaymentID = 501,

CourierID = courier.TrackingNumber,

Amount = 150.75,

PaymentDate = DateTime.Today

};

Console.WriteLine(payment);

Console.WriteLine("\nCourier Company Object:");

CourierCompany company = new CourierCompany();

company.companyName = "Hexaware Courier Co.";

company.courierDetails = new Courier[] { courier };

company.employeeDetails = new Employee[] { employee };

company.locationDetails = new Location[] { location };

Console.WriteLine($"Company Name: {company.companyName}");

Console.WriteLine("Courier Records:");

foreach (var c in company.courierDetails) Console.WriteLine(" - " + c);

Console.WriteLine("Employees:");

foreach (var e in company.employeeDetails) Console.WriteLine(" - " + e);

Console.WriteLine("Locations:");

foreach (var l in company.locationDetails) Console.WriteLine(" - " + l);

}

}

//Task 6: Interface and Abstract Services

using System;

public interface ICourierUserService

{

long PlaceOrder(Courier courierObj);

string GetOrderStatus(long trackingNumber);

bool CancelOrder(long trackingNumber);

Courier[] GetAssignedOrder(int courierStaffId);

}

public interface ICourierAdminService

{

int AddCourierStaff(Employee obj);

}

public class Courier

{

private static long trackingSeed = 100000;

public long CourierID { get; set; }

public string SenderName { get; set; }

public string SenderAddress { get; set; }

public string ReceiverName { get; set; }

public string ReceiverAddress { get; set; }

public double Weight { get; set; }

public string Status { get; set; }

public long TrackingNumber { get; set; }

public DateTime DeliveryDate { get; set; }

public int UserId { get; set; }

public int CourierStaffId { get; set; }

public Courier()

{

TrackingNumber = ++trackingSeed;

}

public Courier(long courierID, string senderName, string senderAddress, string receiverName,

string receiverAddress, double weight, string status, DateTime deliveryDate,

int userId, int courierStaffId)

{

CourierID = courierID;

SenderName = senderName;

SenderAddress = senderAddress;

ReceiverName = receiverName;

ReceiverAddress = receiverAddress;

Weight = weight;

Status = status;

DeliveryDate = deliveryDate;

UserId = userId;

CourierStaffId = courierStaffId;

TrackingNumber = ++trackingSeed;

}

public override string ToString()

{

return $"Courier {TrackingNumber}: {SenderName} ➔ {ReceiverName}, Staff: {CourierStaffId}, Status: {Status}";

}

}

public class Employee

{

public int EmployeeID { get; set; }

public string EmployeeName { get; set; }

public string Email { get; set; }

public string ContactNumber { get; set; }

public string Role { get; set; }

public double Salary { get; set; }

public Employee(int id, string name, string email, string contact, string role, double salary)

{

EmployeeID = id;

EmployeeName = name;

Email = email;

ContactNumber = contact;

Role = role;

Salary = salary;

}

public override string ToString()

{

return $"{EmployeeID} - {EmployeeName} ({Role})";

}

}

public class CourierUserServiceImpl : ICourierUserService

{

private static Courier[] courierList = new Courier[100];

private static int count = 0;

public long PlaceOrder(Courier courierObj)

{

if (count < courierList.Length)

{

courierList[count++] = courierObj;

return courierObj.TrackingNumber;

}

return -1;

}

public string GetOrderStatus(long trackingNumber)

{

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

{

if (courierList[i].TrackingNumber == trackingNumber)

return courierList[i].Status;

}

return "Tracking number not found.";

}

public bool CancelOrder(long trackingNumber)

{

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

{

if (courierList[i].TrackingNumber == trackingNumber && courierList[i].Status.ToLower() == "processing")

{

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

{

courierList[j] = courierList[j + 1];

}

count--;

return true;

}

}

return false;

}

public Courier[] GetAssignedOrder(int courierStaffId)

{

int found = 0;

Courier[] temp = new Courier[count];

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

{

if (courierList[i].CourierStaffId == courierStaffId)

temp[found++] = courierList[i];

}

Courier[] result = new Courier[found];

for (int i = 0; i < found; i++)

{

result[i] = temp[i];

}

return result;

}

}

public class CourierAdminServiceImpl : CourierUserServiceImpl, ICourierAdminService

{

private static Employee[] employeeList = new Employee[100];

private static int empCount = 0;

private static int nextId = 100;

public int AddCourierStaff(Employee obj)

{

if (empCount < employeeList.Length)

{

obj.EmployeeID = nextId++;

employeeList[empCount++] = obj;

return obj.EmployeeID;

}

return -1;

}

}

class Program

{

static void Main()

{

Console.WriteLine("=== Testing Task 6 Service Interfaces (Array Based) ===");

ICourierUserService userService = new CourierUserServiceImpl();

Courier newCourier = new Courier(101, "Alice", "Chennai", "Bob", "Delhi", 5.0,

"Processing", DateTime.Now.AddDays(3), 1, 200);

long trackingNumber = userService.PlaceOrder(newCourier);

Console.WriteLine("Placed Courier: " + trackingNumber);

Console.WriteLine("Status: " + userService.GetOrderStatus(trackingNumber));

bool isCancelled = userService.CancelOrder(trackingNumber);

Console.WriteLine("Cancelled: " + isCancelled);

ICourierAdminService adminService = new CourierAdminServiceImpl();

int empId = adminService.AddCourierStaff(new Employee(0, "Sam", "sam@hexaware.com", "9988776655", "Delivery", 15000));

Console.WriteLine("Added Employee ID: " + empId);

Console.WriteLine("Orders Assigned to Staff ID 200:");

var assigned = userService.GetAssignedOrder(200);

foreach (var c in assigned)

{

Console.WriteLine(c);

}

}

}

Task 7: Exception Handling (Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally,thow & throws keyword usage)

Task 8: Collections Scope: ArrayList/Hashmap Task: Improve the Courier Management System by using Java collections:

Task 8: Service implementation

Task 9: Database Interaction

These topics are not covered yet!!