DAY 6  
OOPS CONCEPT

CODING:

**Program.cs**

using RequestTrackerAppModelLib;

using RequestTrackerModelLib;

using RequestTrackerModelLibrary;

using System.Globalization;

namespace RequestTrackerApp

{

internal class Program

{

Employee[] employees;

public Program()

{

employees = new Employee[3];

}

void PrintMenu()

{

Console.WriteLine("1. Add Employee");

Console.WriteLine("2. Print Employees");

Console.WriteLine("3. Search Employee by ID");

Console.WriteLine("0. Exit");

}

void EmployeeInteraction()

{

int choice = 0;

do

{

PrintMenu();

Console.WriteLine("Please select an option");

choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 0:

Console.WriteLine("Bye.....");

break;

case 1:

AddEmployee();

break;

case 2:

PrintAllEmployees();

break;

case 3:

SearchAndPrintEmployee();

break;

default:

Console.WriteLine("Invalid choice. Try again");

break;

}

} while (choice != 0);

}

void AddEmployee()

{

if (employees[employees.Length - 1] != null)

{

Console.WriteLine("Sorry we have reached the maximum number of employees");

return;

}

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

{

if (employees[i] == null)

{

employees[i] = CreateEmployee(i);

}

}

}

void PrintAllEmployees()

{

if (employees[0] == null)

{

Console.WriteLine("No Employees available");

return;

}

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

{

if (employees[i] != null)

{

Company company = new Company();

company.EmployeeClientVisit(employees[i]);

PrintEmployee(employees[i]);

}

}

}

Employee CreateEmployee(int id)

{

Employee employee = new Employee();

Console.WriteLine("Please enter the type of employee");

string type = Console.ReadLine();

if (type == "Permanent")

employee = new PermanentEmployee();

else if (type == "Contract")

employee = new ContractEmployee();

employee.Id = 101 + id;

employee.BuildEmployeeFromConsole();

return employee;

}

void PrintEmployee(Employee employee)

{

Console.WriteLine("---------------------------");

Console.WriteLine(employee);

Console.WriteLine("---------------------------");

}

int GetIdFromConsole()

{

int id = 0;

Console.WriteLine("Please enter the employee Id");

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Invalid entry. Please try again");

}

return id;

}

void SearchAndPrintEmployee()

{

Console.WriteLine("Print One employee");

int id = GetIdFromConsole();

Employee employee = SearchEmployeeById(id);

if (employee == null)

{

Console.WriteLine("No such Employee is present");

return;

}

PrintEmployee(employee);

}

Employee SearchEmployeeById(int id)

{

Employee employee = null;

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

{

// if ( employees[i].Id == id && employees[i] != null)//Will lead to exception

if (employees[i] != null && employees[i].Id == id)

{

employee = employees[i];

break;

}

}

return employee;

}

static void Main(string[] args)

{

Program program = new Program();

program.EmployeeInteraction();

ContractEmployee employee = new ContractEmployee();

employee.BuildEmployeeFromConsole();

Console.WriteLine(employee);

}

}

}

**Company.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public class Company

{

public void EmployeeClientVisit(IClientInteraction clientInteraction)

{

clientInteraction.GetPayment();

}

}

}

**ContractEmployee.cs**

using RequestTrackerModelLib;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerModelLibrary

{

public class ContractEmployee : Employee

{

public double WagesPerDay { get; set; }

public ContractEmployee()

{

WagesPerDay = 0;

Type = "ContractEmployee";

Console.WriteLine("Contract employee constructor");

}

public ContractEmployee(int id, string name, DateTime dateOfBirth, double salary, double wagesPerDay) : base(id, name, dateOfBirth, salary)

{

Console.WriteLine("Contract Employee class prameterized constructor");

WagesPerDay = wagesPerDay;

}

public override void BuildEmployeeFromConsole()

{

base.BuildEmployeeFromConsole();

Console.WriteLine("Please enter the Per Day Wage");

WagesPerDay = Convert.ToDouble(Console.ReadLine());

CalculateSalary();

}

private void CalculateSalary()

{

Salary = WagesPerDay \* 30;

}

public override void PrintEmployeeDetails()

{

base.PrintEmployeeDetails();

Console.WriteLine("Wages/Day : " + WagesPerDay);

}

}

}

**Department.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

public int Department\_Head { get; set; }}}

**Employee.cs**

using RequestTrackerAppModelLib;

namespace RequestTrackerModelLib

{

public class Employee : IClientInteraction, IInternalCompanyWorking

{

public Department EmployeeDepartment { get; set; }

int age;

DateTime dob;

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public int Age

{

get

{

return age;

}

}

public DateTime DateOfBirth

{

get => dob;

set

{

dob = value;

age = ((DateTime.Today - dob).Days) / 365;

}

}

public double Salary { get; set; }

public string Type { get; set; }

public Employee()

{

Console.WriteLine("Employee class default constructor");

Id = 0;

Name = string.Empty;

Salary = 0.0;

DateOfBirth = new DateTime();

Type = string.Empty;

}

public Employee(int id, string name, DateTime dateOfBirth, double salary)

{

Console.WriteLine("Employee class prameterized constructor");

Id = id;

Name = name;

DateOfBirth = dateOfBirth;

}

public virtual void BuildEmployeeFromConsole()

{

Console.WriteLine("Please enter the Name");

Name = Console.ReadLine() ?? String.Empty;

Console.WriteLine("Please enter the Date of birth");

DateOfBirth = Convert.ToDateTime(Console.ReadLine());

}

public virtual void PrintEmployeeDetails()

{

Console.WriteLine("Employee Type : " + Type);

Console.WriteLine("Employee Id : " + Id);

Console.WriteLine("Employee Name " + Name);

Console.WriteLine("Date of birth : " + DateOfBirth);

Console.WriteLine("Employee Age : " + Age);

}

public override string ToString()

{

return "Employee Type : " + Type

+ "\nEmployee Id : " + Id

+ "\nEmployee Name " + Name

+ "\nDate of birth : " + DateOfBirth

+ "\nEmployee Age : " + Age;

}

public void GetOrder()

{

Console.WriteLine("Order fetched by " + Name);

}

public void GetPayment()

{

Console.WriteLine("Get the payment as per terms");

}

public void RaiseRequest()

{

throw new NotImplementedException();

}

public void CloseRequest()

{

throw new NotImplementedException();

}

}

}

**IClientInteraction.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public interface IClientInteraction

{

void GetOrder();

void GetPayment();

}

}

**PermanentEmployee.cs**

using RequestTrackerModelLib;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public class PermanentEmployee:Employee

{

public PermanentEmployee()

{

Type = "PermanentEmployee";

}

public PermanentEmployee(int id, string name, DateTime dateOfBirth, double salary)

{

Salary = salary;

}

public override void BuildEmployeeFromConsole()

{

base.BuildEmployeeFromConsole();

Console.WriteLine("Please enter the Basic Salary");

Salary = Convert.ToDouble(Console.ReadLine());

}

public override void PrintEmployeeDetails()

{

base.PrintEmployeeDetails();

Console.WriteLine("Employee Salary : Rs." + Salary);

}

public void SpecialPermanentMethod()

{

Console.WriteLine("Special permanent method");

}

}

}

**IInternalCompanyWorking.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public interface IInternalCompanyWorking

{

void RaiseRequest();

void CloseRequest();

}

}

OUTPUT SCREENSHOTS



