**Programming task: Employee Payroll System (C# console application)**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace payrollSystem

{

**// BaseEmployee class**

public class BaseEmployee

{

public string Name { get; set; }

public int ID { get; set; }

public string Role { get; set; }

public double BasicPay { get; set; }

public double Allowances { get; set; }

public double Deductions { get; set; }

**// Constructor**

public BaseEmployee(string name, int id, string role, double basicPay, double allowances, double deductions)

{

Name = name;

ID = id;

Role = role;

BasicPay = basicPay;

Allowances = allowances;

Deductions = deductions;

}

**// Method to calculate salary**

public virtual double CalculateSalary()

{

return BasicPay + Allowances - Deductions;

}

**// Method to display employee details**

public virtual void DisplayDetails()

{

Console.WriteLine($"ID: {ID}, Name: {Name}, Role: {Role}, Salary: {CalculateSalary():C2}");

}

}

**// Derived class: Manager**

public class Manager : BaseEmployee

{

public double Bonus { get; set; }

public Manager(string name, int id, double basicPay, double allowances, double deductions, double bonus)

: base(name, id, "Manager", basicPay, allowances, deductions)

{

Bonus = bonus;

}

public override double CalculateSalary()

{

return base.CalculateSalary() + Bonus;

}

public override void DisplayDetails()

{

base.DisplayDetails();

Console.WriteLine($"Bonus: {Bonus:C2}");

}

}

**// Derived class: Developer**

public class Developer : BaseEmployee

{

public Developer(string name, int id, double basicPay, double allowances, double deductions)

: base(name, id, "Developer", basicPay, allowances, deductions)

{

}

}

**// Derived class: Intern**

public class Intern : BaseEmployee

{

public Intern(string name, int id, double basicPay, double allowances, double deductions)

: base(name, id, "Intern", basicPay, allowances, deductions)

{

}

}

class Program

{

static List<BaseEmployee> employees = new List<BaseEmployee>();

static string filePath = "employees.txt";

static void Main(string[] args)

{

LoadEmployeeData(); // Load employee data from file on startup

while (true)

{

Console.WriteLine("\n--- Employee Payroll System ---");

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

Console.WriteLine("2. Display All Employees");

Console.WriteLine("3. Calculate and Display Individual Salaries");

Console.WriteLine("4. Calculate Total Payroll");

Console.WriteLine("5. Save Employee Data");

Console.WriteLine("6. Exit");

Console.Write("Enter your choice: ");

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

switch (choice)

{

case 1:

AddEmployee();

break;

case 2:

DisplayAllEmployees();

break;

case 3:

CalculateIndividualSalaries();

break;

case 4:

CalculateTotalPayroll();

break;

case 5:

SaveEmployeeData();

break;

case 6:

Console.WriteLine("Exiting the system.");

return;

default:

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

break;

}

}

}

static void AddEmployee()

{

Console.Write("Enter Name: ");

string name = Console.ReadLine();

Console.Write("Enter ID: ");

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

Console.Write("Enter Role (Manager/Developer/Intern): ");

string role = Console.ReadLine();

Console.Write("Enter Basic Pay: ");

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

Console.Write("Enter Allowances: ");

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

Console.Write("Enter Deductions: ");

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

if (role.ToLower() == "manager")

{

Console.Write("Enter Bonus: ");

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

employees.Add(new Manager(name, id, basicPay, allowances, deductions, bonus));

}

else if (role.ToLower() == "developer")

{

employees.Add(new Developer(name, id, basicPay, allowances, deductions));

}

else if (role.ToLower() == "intern")

{

employees.Add(new Intern(name, id, basicPay, allowances, deductions));

}

else

{

Console.WriteLine("Invalid role. Employee not added.");

}

Console.WriteLine("Employee added successfully.");

}

static void DisplayAllEmployees()

{

if (employees.Count == 0)

{

Console.WriteLine("No employees found.");

return;

}

Console.WriteLine("\n--- Employee Details ---");

foreach (var emp in employees)

{

emp.DisplayDetails();

}

}

static void CalculateIndividualSalaries()

{

if (employees.Count == 0)

{

Console.WriteLine("No employees to calculate salary.");

return;

}

Console.WriteLine("\n--- Salary Details ---");

foreach (var emp in employees)

{

Console.WriteLine($"Salary for {emp.Name} (ID: {emp.ID}): {emp.CalculateSalary():C2}");

}

}

static void CalculateTotalPayroll()

{

if (employees.Count == 0)

{

Console.WriteLine("No employees found.");

return;

}

double totalPayroll = 0;

foreach (var emp in employees)

{

totalPayroll += emp.CalculateSalary();

}

Console.WriteLine($"Total Payroll: {totalPayroll:C2}");

}

**// Save employee data to file**

static void SaveEmployeeData()

{

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (var emp in employees)

{

writer.WriteLine($"{emp.Name},{emp.ID},{emp.Role},{emp.BasicPay},{emp.Allowances},{emp.Deductions}");

if (emp is Manager manager)

{

writer.WriteLine($"Bonus: {manager.Bonus}");

}

}

}

Console.WriteLine("Employee data saved successfully.");

}

**// Load employee data from file**

static void LoadEmployeeData()

{

if (File.Exists(filePath))

{

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

var details = line.Split(',');

string name = details[0];

int id = int.Parse(details[1]);

string role = details[2];

double basicPay = double.Parse(details[3]);

double allowances = double.Parse(details[4]);

double deductions = double.Parse(details[5]);

if (role == "Manager")

{

double bonus = double.Parse(reader.ReadLine().Split(':')[1].Trim());

employees.Add(new Manager(name, id, basicPay, allowances, deductions, bonus));

}

else if (role == "Developer")

{

employees.Add(new Developer(name, id, basicPay, allowances, deductions));

}

else if (role == "Intern")

{

employees.Add(new Intern(name, id, basicPay, allowances, deductions));

}

}

}

}

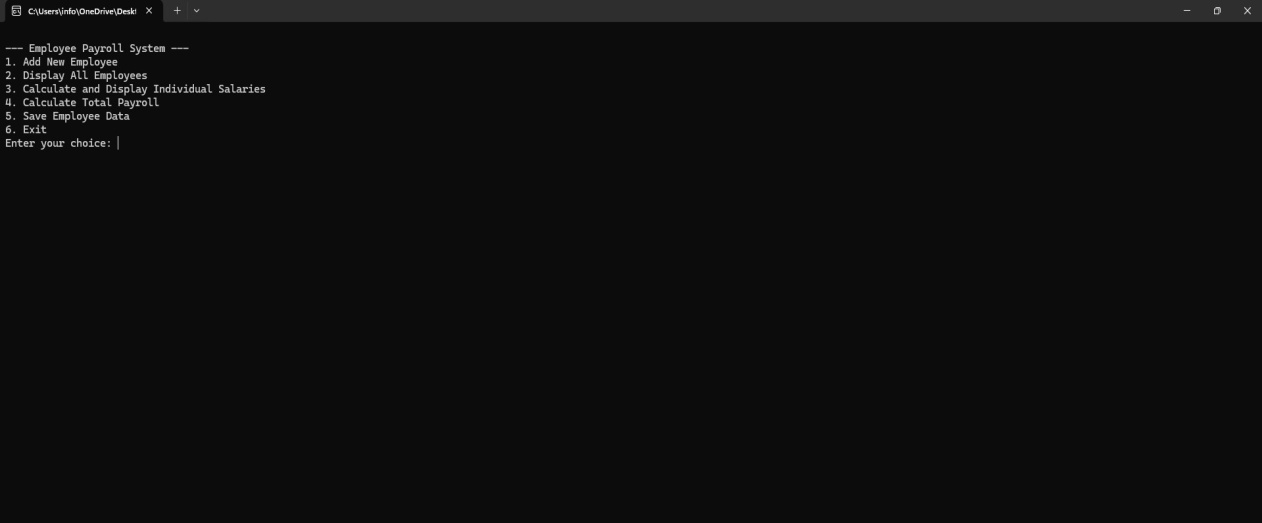
}

}

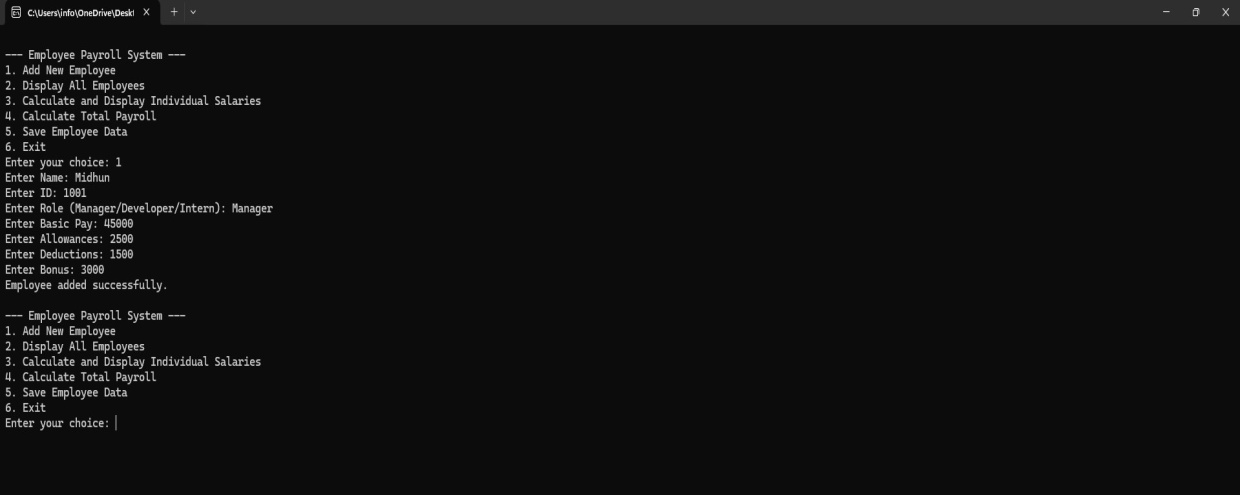
}

**The output of the above showing code is given below:**

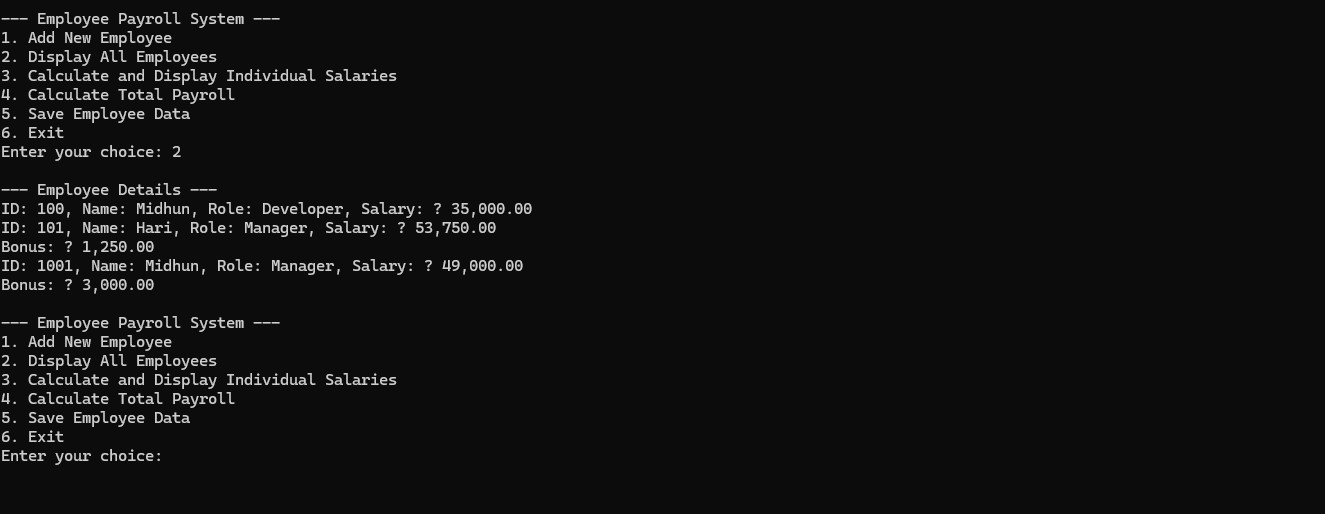
1. **Employee Payroll System**

****

1. **Add New Employee**

****

1. **Display All Employees**

****

1. **Calculate and Display Individual Salaries**

****

1. **Calculate Total Payroll**

****

1. **Save Employee Data**

****