12214994_SaurabhRana

Question No : 1 / 1

Title: Employee Salary Calculator

All the classes must be in public.

Description:

You are developing a payroll system that calculates the salaries of different types of employees. The program uses a hierarchy of classes to represent various employee types and their corresponding salary calculation methods. Your task is to implement the missing parts of the program to ensure it correctly computes the salaries based on employee type and hours worked. Write the solution within the Program.cs file.

The program consists of the following classes:

- **Employee:** The base class representing an employee. It contains a virtual method `CalculateSalary()` which computes and returns the salary of the employee based on their hourly rate and hours worked.
- **FullTimeEmployee:** A derived class from Employee representing a full-time employee. It overrides the `CalculateSalary()` method to compute and return the salary of the full-time employee, which is their hourly rate multiplied by hours worked.
- **PartTimeEmployee:** A derived class from Employee representing a part-time employee. It overrides the `CalculateSalary()` method to compute and return the salary of the part-time employee, which is their hourly rate multiplied by hours worked, with a **20% discount** applied to the rate.
- **Intern:** A derived class from Employee representing an intern. It overrides the `CalculateSalary()` method to compute and return the salary of the intern, which is their hourly rate multiplied by hours worked, with a **40% discount** applied to the rate.
- **Program:** The main program class containing the `Main()` method. It prompts the user to input the number of employees and their details (type, hourly rate, and hours worked). It then creates an array of Employee objects based on the user input and calculates the salary of each employee.

Tasks:

1. Complete the implementation of the `Main()` method to prompt the user to input the number of employees they want to calculate the salaries for. Ensure that the input is a valid positive integer.
2. Implement the logic inside the `for` loop to create instances of `FullTimeEmployee`, `PartTimeEmployee`, or `Intern` based on the user input, and store them in the employees array accordingly.
3. Ensure that after creating the array of employees, the program calculates and outputs the salary of each employee stored in the array.
4. If the user enters an invalid number of employees (zero or negative), display the message **"Please enter a valid positive integer for the number of employees."**

Handle exceptions:

Ensure that the program handles exceptions for invalid user inputs, such as non-numeric values for the hourly rate or hours worked. If such an invalid input is detected, the program should prompt the user to re-enter the details for the current employee, displaying the following error messages:

1. If the hourly rate input is invalid (non-numeric), display:
   "Invalid input for hourly rate. Please enter a numeric value."
2. If the hours worked input is invalid (non-numeric), display:
   "Invalid input for hours worked. Please enter a numeric value."
3. If the employee type input is invalid, display:
   "Unknown employee type. Please enter FullTimeEmployee, PartTimeEmployee, or Intern."

Constraints:

- Ensure error handling for invalid user inputs.
- Use appropriate inheritance and method overriding concepts to achieve the desired behavior.
- Implement the salary calculation logic efficiently to handle varying employee types and work hours.

Input Format:

The first line of input should be number of employees in integer.

For each employee:

- **fulltimeemployee:**
    - The user inputs a double-precision floating-point representing the hourlyRate.
    - The user inputs a double-precision floating-point representing the hoursWorked.
- **parttimeemployee:**
    - The user inputs a double-precision floating-point representing the hourlyRate.
    - The user inputs a double-precision floating-point representing the hoursWorked.
- **intern:**
    - The user inputs a double-precision floating-point representing the hourlyRate.
    - The user inputs a double-precision floating-point representing the hoursWorked.

Output Format:

If number of employees > 0:

- The first line of output should be **"Salaries of the employees:"**
- The next lines of output should be displayed as salary of each employee based on input.

If number of employees <= 0:

- The first line of output should be **"Please enter a valid positive integer."**

The output should be in the format of sample outputs.

Sample Input 1

```
3
fulltimeemployee
1000
2
parttimeemployee
1000
2
intern
1000
2
```

Sample Output 1

```
Salaries of the employees:
Salary of Employee 1 (FullTimeEmployee): 2000
Salary of Employee 2 (PartTimeEmployee): 1600
Salary of Employee 3 (Intern): 1200
```

Sample Input 2

```
0
```

Sample Output 2

```
Please enter a valid positive integer.
```

Commands to Run the Project:

- `cd dotnetapp`
- Select the dotnet project folder
- `dotnet run`
- `dotnet build`

Answer:

```
class Employee

{



   protected double hourRate,hours;

   public Employee(double hourRate,double hours)

   {

      this.hourRate=hourRate;

      this.hours=hours;

   }

   public virtual double CalculateSalary()

   {

      return hourRate * hours;

   }

}

class FullTimeEmp : Employee

{
    public FullTimeEmp(double hourRate, double hours) : base(hourRate, hours) { }

    public override double CalculateSalary()
    {
        return hourRate * hours;
    }
}
class Intern : Employee
{
    public Intern(double hourRate, double hours) : base(hourRate, hours) { }

    public override double CalculateSalary()
    {
        return hourRate * 0.6 * hours;
    }
}
class PartTimeEmp : Employee
{
    public PartTimeEmp(double hourRate, double hours) : base(hourRate, hours) { }

    public override double CalculateSalary()
    {
        return hourRate * 0.8 * hours;
    }
}
```

```
class Program

{

    static void Main()

    {

        Console.Write("Enter the number of Employees: ");

        int number;

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

        {

            Console.Write("Please enter a valid integer :");

        }


        Employee[] emp = new Employee[number];

        string[] empTypes = new string[number];

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

        {

            while (true)

            {

                double hourRate,hours;

                Console.Write("Enter employee type: ");

                string empType = (Console.ReadLine() ?? "").ToLower();

                if (empType == "fulltimeemployee")

                {

                    Console.Write("Enter hour rate: ");

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

                    Console.Write("Enter hours: ");
                    hours = Convert.ToDouble(Console.ReadLine());

                    emp[i] = new FullTimeEmp(hourRate,hours);

                    empTypes[i] = "FullTimeEmployee";

                    break;


                }

                else if (empType == "parttimeemployee")

                {

                    Console.Write("Enter hour rate: ");

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

                    Console.Write("Enter hours: ");

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

                    emp[i] = new PartTimeEmp(hourRate, hours);

                    empTypes[i] = "PartTimeEmployee";

                    break;

                }

                else if (empType == "intern")

                {

                    Console.Write("Enter hour rate: ");

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

                    Console.Write("Enter hours: ");

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

                    emp[i] = new Intern(hourRate, hours);

                    empTypes[i] = "Intern";

                    break;
```

```
            }
            else
            {
                Console.WriteLine("Unknown employee type. Please enter FullTimeEmployee,
PartTimeEmployee, or Intern.");
            }
        }
    }
    Console.WriteLine("Salaries of the employees:");
    for (int i = 0; i < number; i++)
    {
        Console.WriteLine($"Salary of Employee {i + 1} ({empTypes[i]}): {emp[i].CalculateSalary()}");
    }
  }
}
```