**K. Vamsi Krishna Prasad**

**- J00744483**

**Tutorial 1: Creating SubClasses**

**Code:**

**Employee.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class Employee

{

public string Name { get; set; }

public Employee(string name)

{

this.Name = name; }

}

}

**SalariedEmployee.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class SalariedEmployee : Employee

{

public decimal AnnualSalary { get; set; }

public SalariedEmployee(string name, decimal salary) : base(name)

{

this.AnnualSalary = salary;

}

}

}

**HourlyEmployee.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

internal class HourlyEmployee: Employee

{

public decimal PayRate { get; set; }

private int minutesWorked;

public HourlyEmployee(string name, decimal payRate) : base(name)

{

this.PayRate = payRate;

this.minutesWorked = 0;

}

public void AddTime(int minutes)

{

this.minutesWorked += minutes;

}

public decimal Hoursworked()

{

return minutesWorked / 60.0m;

}

}

}

**Program.cs**:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Employee emp1 = new Employee("Spongebob");

SalariedEmployee salEmp2 = new SalariedEmployee("Clark Kent", 45000);

HourlyEmployee hourlyEmp3 = new HourlyEmployee("Batman", 40.0m);

hourlyEmp3.AddTime(30);

decimal hours = hourlyEmp3.Hoursworked();

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Output:

**Tutorial 2: Implementing the Gross Pay Method**

**Code:**

**Employee.cs:**

public abstract class Employee

{

public string Name { get; set; }

public Employee(string name)

{

this.Name = name;

}

public abstract decimal GrossPay();

}

**HourlyEmployee.cs:**

class HourlyEmployee: Employee

{

public decimal PayRate { get; set; }

private int minutesWorked;

public HourlyEmployee(string name, decimal payRate) : base(name)

{

this.PayRate = payRate;

this.minutesWorked = 0;

}

public void AddTime(int minutes)

{

this.minutesWorked += minutes;

}

public decimal Hoursworked()

{

return minutesWorked / 60.0m;

}

public override decimal GrossPay()

{

return Hoursworked() \* PayRate;

}

}

**SalariedEmployee.cs:**

class SalariedEmployee : Employee

{

public decimal AnnualSalary { get; set; }

public SalariedEmployee(string name, decimal salary) : base(name)

{

this.AnnualSalary = salary;

}

public override decimal GrossPay()

{

return AnnualSalary / 52 \* 2;

}

}

**Program.cs:**

static void Main()

{

Random r = new Random();

List<Employee> employees = new List<Employee>();

string[] employeeNames = {"Superman", "Green Lantern","Batman","Spiderman",

"Thor", "Hal Jordan", "Wonder Woman", "Captain America"};

foreach(string name in employeeNames)

{

Employee emp = null;

int SalOrHr1 = r.Next(0, 2);

if(SalOrHr1 == 0)

{

decimal payRate = (decimal)r.NextDouble() \* 20 + 10;

HourlyEmployee hourly = new HourlyEmployee(name, payRate);

hourly.AddTime(r.Next(30, 480));

emp = hourly;

}

else

{

decimal salary = r.Next(40000, 80000);

emp = new SalariedEmployee(name, salary);

}

employees.Add(emp);

}

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new EmployeeInheritanceForm(employees));

}

**EmployeeInheritanceForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

public partial class EmployeeInheritanceForm : Form

{

List<Employee> employees;

public EmployeeInheritanceForm()

{

this.employees = employees;

InitializeComponent();

lblDetails.Text = employeeDetails();

}

private decimal totalGrossPay()

{

decimal result = 0;

foreach(Employee e in employees)

{

result += e.GrossPay();

}

return result;

}

private decimal averageGrossPay()

{

return totalGrossPay() / employees.Count;

}

private string employeeDetails()

{

string result = "";

foreach(Employee e in employees)

{

result += string.Format("Nmae: {0}, Gross Pay: {1:c}\n", e.Name, e.GrossPay());

}

return result;

}

}

}

**Output:**

A screenshot of a computer screen

Description automatically generated

**Tutorial 3: Dynamic Dispatch and the ToString Method**

**Code:**

**Employee.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

public abstract class Employee

{

public string Name { get; set; }

public Employee(string name)

{

this.Name = name;

}

public abstract decimal GrossPay();

public override string ToString()

{

return "Name: " + Name;

}

}

}

**HourlyEmployee.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class HourlyEmployee : Employee

{

public decimal PayRate { get; set; }

private int minutesWorked;

public HourlyEmployee(string name, decimal payRate) : base(name)

{

this.PayRate = payRate;

this.minutesWorked = 0;

}

public void AddTime(int minutes)

{

this.minutesWorked += minutes;

}

public decimal HoursWorked()

{

return minutesWorked / 60.0m;

}

public override decimal GrossPay()

{

return HoursWorked() \* PayRate;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(" . Hourly pay rate: {0:c}", PayRate);

return result;

}

}

}

**EmployeeInheritanceForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

public partial class EmployeeInheritanceForm : Form

{

List<Employee> employees;

public EmployeeInheritanceForm(List<Employee> employees)

{

this.employees = employees;

InitializeComponent();

lblDetails.Text = employeeDetails();

lblNumEmployees.Text = employees.Count.ToString();

lblTotalGrossPay.Text = string.Format("{0:c}", totalGrossPay());

lblAverageGrossPay.Text = string.Format("{0:c}", averageGrossPay());

lstEmployees.Items.AddRange(employees.ToArray());

}

private decimal totalGrossPay()

{

decimal result = 0;

foreach(Employee e in employees)

{

result += e.GrossPay();

}

return result;

}

private decimal averageGrossPay()

{

return totalGrossPay() / employees.Count;

}

private string employeeDetails()

{

string result = "";

foreach(Employee e in employees)

{

//result += String.Format("Name: {0}, Gross Pay: {1:c}\n", e.Name, e.GrossPay());

result += e.ToString() + "\n";

}

return result;

}

}

}

**Output:**

A screenshot of a computer

Description automatically generated

**Tutorial 4: Interfaces**

**Code:**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Random r = new Random();

List<Employee> employees = new List<Employee>();

List<ITaxable> taxEmployees = new List<ITaxable>();

string[] employeeNames = {"Superman", "Green Lantern", "Batman", "Spiderman", "Thor",

"Hal Jordan", "Wonder Woman", "Captain American"};

foreach(string name in employeeNames)

{

Employee emp = null;

int salOrHrl = r.Next(0, 2);

if (salOrHrl == 0)

{

decimal payRate = (decimal)r.NextDouble() \* 20 + 10;

HourlyEmployee hourly = new HourlyEmployee(name, payRate);

hourly.AddTime(r.Next(30, 480));

emp = hourly;

}

else if(salOrHrl == 1)

{

decimal salary = r.Next(40000, 80000);

emp = new SalariedEmployee(name, salary);

taxEmployees.Add((ITaxable)emp);

}

else if(salOrHrl == 2)

{

decimal contractAmount = (decimal)r.NextDouble() \* 300 + 100;

emp = new ContractEmployee(name, contractAmount);

}

employees.Add(emp);

}

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new EmployeeInheritanceForm(employees, taxEmployees));

}

}

}

ContractEmployee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class ContractEmployee : Employee

{

public decimal ContractAmount { get; set; }

public ContractEmployee(string name, decimal contratAmount) : base(name)

{

this.ContractAmount = contratAmount;

}

public override decimal GrossPay()

{

return ContractAmount;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(".Contract amount: {0:c}", ContractAmount);

return result;

}

}

}

ITaxable.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

public interface ITaxable

{

decimal CalculateTax();

}

}

EmployeeInheritanceForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

public partial class EmployeeInheritanceForm : Form

{

private List<Employee> employees;

private List<ITaxable> taxEmployees;

public EmployeeInheritanceForm(List<Employee> employees, List<ITaxable> taxEmployees)

{

this.employees = employees;

this.taxEmployees = taxEmployees;

InitializeComponent();

lblDetails.Text = employeeDetails();

lblNumEmployees.Text = employees.Count.ToString();

lblTotalGrossPay.Text = string.Format("{0:c}", totalGrossPay());

lblAverageGrossPay.Text = string.Format("{0:c}", averageGrossPay());

lstEmployees.Items.AddRange(employees.ToArray());

lblTotalTax.Text = totalTax().ToString();

}

private decimal totalGrossPay()

{

decimal result = 0;

foreach(Employee e in employees)

{

result += e.GrossPay();

}

return result;

}

private decimal averageGrossPay()

{

return totalGrossPay() / employees.Count;

}

private string employeeDetails()

{

string result = "";

foreach(Employee e in employees)

{

//result += String.Format("Name: {0}, Gross Pay: {1:c}\n", e.Name, e.GrossPay());

result += e.ToString() + "\n";

}

return result;

}

private decimal totalTax()

{

decimal result = 0;

foreach(ITaxable e in taxEmployees)

{

result += e.CalculateTax();

}

return result;

}

}

}

HourlyEmployee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class HourlyEmployee : Employee, ITaxable

{

public decimal PayRate { get; set; }

private int minutesWorked;

public HourlyEmployee(string name, decimal payRate) : base(name)

{

this.PayRate = payRate;

this.minutesWorked = 0;

}

public void AddTime(int minutes)

{

this.minutesWorked += minutes;

}

public decimal HoursWorked()

{

return minutesWorked / 60.0m;

}

public override decimal GrossPay()

{

return HoursWorked() \* PayRate;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(" . Hourly pay rate: {0:c}", PayRate);

return result;

}

public decimal CalculateTax()

{

return (PayRate < 20) ? GrossPay() \* 0.3m : GrossPay() \* 0.4m;

}

}

}

SalariedEmployee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class SalariedEmployee : Employee, ITaxable

{

public decimal AnnualSalary { get; set; }

public override decimal GrossPay()

{

return AnnualSalary / 52 \* 2;

// Gross pay is simply two weeks of the annual salary (52 weeks of year)

}

public SalariedEmployee(string name, decimal salary) : base(name)

{

this.AnnualSalary = salary;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(" . Annual Salary: {0:c}",AnnualSalary);

return result;

}

public decimal CalculateTax()

{

if (AnnualSalary > 65000)

{

return AnnualSalary \* 45 /100;

}

else

{

return AnnualSalary \* 35 / 100;

}

}

}

}

Employee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

public abstract class Employee

{

public string Name { get; set; }

public Employee(string name)

{

this.Name = name;

}

public abstract decimal GrossPay();

public override string ToString()

{

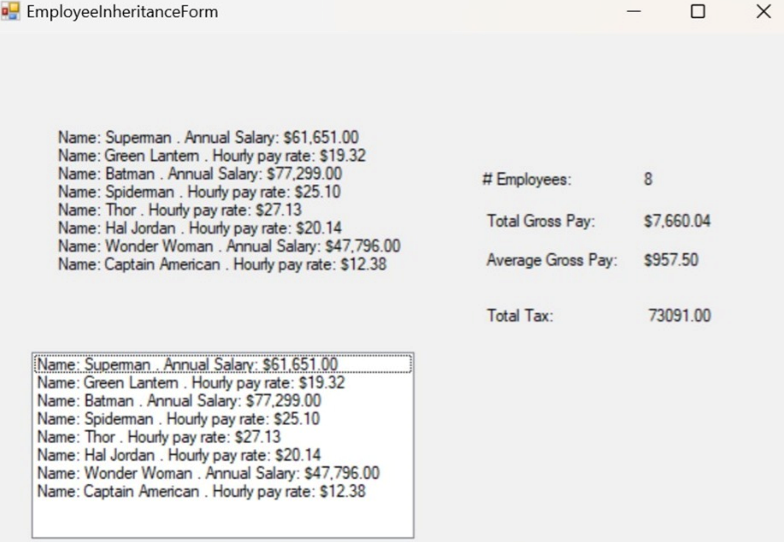
return "Name: " + Name;

}

}

}

**Output:**

****

**Tutorial 5: Sorting Employees**

**Code:**

Employee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

public abstract class Employee : IComparable<Employee>

{

public string Name { get; set; }

//public string Name1 { get; set; }

public Employee(string name)

{

this.Name = name;

//this.Name1 = name1;

}

public abstract decimal GrossPay();

public override string ToString()

{

return "Name: " + Name;

}

public int CompareTo(Employee other)

{

return this.Name.CompareTo(other.Name);

}

}

}

SalariedEmployee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class SalariedEmployee : Employee, ITaxable

{

public decimal AnnualSalary { get; set; }

public override decimal GrossPay()

{

return AnnualSalary / 52 \* 2;

// Gross pay is simply two weeks of the annual salary (52 weeks of year)

}

public SalariedEmployee(string name, decimal salary) : base(name)

{

this.AnnualSalary = salary;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(" . Annual Salary: {0:c}",AnnualSalary);

return result;

}

public decimal CalculateTax()

{

if (AnnualSalary > 65000)

{

return AnnualSalary \* 45 /100;

}

else

{

return AnnualSalary \* 35 / 100;

}

}

}

}

HourlyEmployee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class HourlyEmployee : Employee, ITaxable

{

public decimal PayRate { get; set; }

private int minutesWorked;

public HourlyEmployee(string name, decimal payRate) : base(name)

{

this.PayRate = payRate;

this.minutesWorked = 0;

}

public void AddTime(int minutes)

{

this.minutesWorked += minutes;

}

public decimal HoursWorked()

{

return minutesWorked / 60.0m;

}

public override decimal GrossPay()

{

return HoursWorked() \* PayRate;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(" . Hourly pay rate: {0:c}", PayRate);

return result;

}

public decimal CalculateTax()

{

return (PayRate < 20) ? GrossPay() \* 0.3m : GrossPay() \* 0.4m;

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Random r = new Random();

List<Employee> employees = new List<Employee>();

List<ITaxable> taxEmployees = new List<ITaxable>();

string[] employeeNames = {"Superman", "Green Lantern", "Batman", "Spiderman", "Thor",

"Hal Jordan", "Wonder Woman", "Captain American"};

foreach(string name in employeeNames)

{

Employee emp = null;

int salOrHrl = r.Next(0, 3);

if (salOrHrl == 0)

{

decimal payRate = (decimal)r.NextDouble() \* 20 + 10;

HourlyEmployee hourly = new HourlyEmployee(name, payRate);

hourly.AddTime(r.Next(30, 480));

emp = hourly;

}

else if(salOrHrl == 1)

{

decimal salary = r.Next(40000, 80000);

emp = new SalariedEmployee(name, salary);

taxEmployees.Add((ITaxable)emp);

}

else if(salOrHrl == 2)

{

decimal contractAmount = (decimal)r.NextDouble() \* 300 + 100;

emp = new ContractEmployee(name, contractAmount);

}

employees.Add(emp);

}

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new EmployeeInheritanceForm(employees, taxEmployees));

}

}

}

NameComparer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class NameComparer : IComparer<Employee>

{

/\*

public int Comparer(Employee x, Employee y)

{

return x.Name.CompareTo(y.Name);

}

\*/

public int Comparer(Employee x, Employee y)

{

return x.GrossPay().CompareTo(y.GrossPay());

}

}

}

EmployeeInheritanceForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Payroll\_Inheritance

{

public partial class EmployeeInheritanceForm : Form

{

private List<Employee> employees;

private List<ITaxable> taxEmployees;

public EmployeeInheritanceForm(List<Employee> employees, List<ITaxable> taxEmployees)

{

this.employees = employees;

this.taxEmployees = taxEmployees;

InitializeComponent();

lblNumEmployees.Text = employees.Count.ToString();

lblTotalGrossPay.Text = string.Format("{0:c}", totalGrossPay());

lblAverageGrossPay.Text = string.Format("{0:c}", averageGrossPay());

lstEmployees.Items.AddRange(employees.ToArray());

lblTotalTax.Text = totalTax().ToString();

lstEmployees.Items.AddRange(employees.ToArray());

}

private decimal totalGrossPay()

{

decimal result = 0;

foreach(Employee e in employees)

{

result += e.GrossPay();

}

return result;

}

private decimal averageGrossPay()

{

return totalGrossPay() / employees.Count;

}

private string employeeDetails()

{

string result = "";

foreach(Employee e in employees)

{

//result += String.Format("Name: {0}, Gross Pay: {1:c}\n", e.Name, e.GrossPay());

result += e.ToString() + "\n";

}

return result;

}

private decimal totalTax()

{

decimal result = 0;

foreach(ITaxable e in taxEmployees)

{

result += e.CalculateTax();

}

return result;

}

private void btnReverse\_Click(object sender, EventArgs e)

{

employees.Reverse();

lstEmployees.Items.Clear();

lstEmployees.Items.Add(employees.ToArray());

}

private void btnSortGrosspay\_Click(object sender, EventArgs e)

{

employees.Sort();

lstEmployees.Items.Clear();

lstEmployees.Items.AddRange(employees.ToArray());

}

private void btnSortName\_Click(object sender, EventArgs e)

{

employees.Sort(new NameComparer());

lstEmployees.Items.Clear();

lstEmployees.Items.AddRange(employees.ToArray());

}

}

}

ITaxable.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

public interface ITaxable

{

decimal CalculateTax();

}

}

ContractEmployee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Payroll\_Inheritance

{

class ContractEmployee : Employee

{

public decimal ContractAmount { get; set; }

public ContractEmployee(string name, decimal contratAmount) : base(name)

{

this.ContractAmount = contratAmount;

}

public override decimal GrossPay()

{

return ContractAmount;

}

public override string ToString()

{

string result = base.ToString();

result += string.Format(".Contract amount: {0:c}", ContractAmount);

return result;

}

}

}

**Output:**

A screenshot of a computer screen

Description automatically generated