Systems Development: Object Oriented Analysis and Design

(H172 35)

Introducing Inheritance

Employees Walkthrough

Step 1 – creating the individual classes

We are going to model 2 different types of employees in a company’s payroll system, commission employees who are paid a percentage of their sales (CommissionEmployee), and base salaried commission employees (BasePlusCommissionEmployee) who receive a base salary plus a percentage of their sales.

We are initially going to implement these two classes independently and then review the redundancy in their implementation to understand how it could be better implemented using an inheritance hierarchy.

1. Referring to the UML diagram below, implement a CommissionEmployee class (CommissionEmployee.cs). Descriptions of how to implement each method are also included below the UML diagram.

CommissionEmployee

- firstName:string

- lastName:string

- socialSecurityNumber:string

- grossSales:decimal

- commissionRate:decimal

+ <<property>> FirstName:string

+ <<property>> LastName:string

+ <<property>> SocialSecurityNumber:string

+ <<property>> GrossSales:decimal

+ <<property>> CommissionRate:decimal

+ CommissionEmployee(first:string, last:string, ssn:string, sales:decimal, rate:decimal)

+ Earnings():decimal

+ ToString():string

Methods for CommissionEmployee class

CommissionEmployees’s public services include a constructor, methods *Earnings()* and *ToString()*, and the public *properties* for manipulating the class’s instance variables. Each of its instance variables is private, and properties will be used to manipulate/access these variables (N.B. get and set methods for properties).

The GrossSales (set) Property performs validation and will throw an exception if the value is < 0 :

if(value >=0)

grossSales = value

else

throw new ArgumentOutOfRangeException(“Gross sales “, value, “GrossSales must be >= 0”);

The ComissionRate (set) Property performs validation and will throw an exception if the value is not in the range > 0 and < 1.

Method *Earnings()* – calculates a CommissionEmployee’s earnings by multiplying the commissionRate by the grossSales and return the result.

Method *ToString()* - one of the methods that every class inherits from class object, which is the root of the C# class hierarchy. It returns a string representing the object, and is called implicitly by an app whenever an object must be converted to a string representation, such as Console’s Write method. Use format specifier ‘C’ to format the grossSales and format specifier ‘F2’ to format the comissionRate with 2 digits of precision to the right of the decimal point.

public **override** string ToString()

{

return string.Format(

“{0}: {1} {2}\n{3}: {4}\n{5}: {6:C}\n{7}: {8:F2}”,

“commission employee”, FirstName, LastName,

“social security number”, SocialSecurityNumber,

“gross sales”, GrossSales, “commission rate”, CommissionRate );

}

1. Create a test class named CommissionEmployeeTest.cs. Inside the Main method, execute the following steps:

* Create a CommissionEmployee object and invoke it’s constructor to initialise it with a set of values (you choose)
* Use the **Properties** to display each attributes value individually to the console and the earnings:

First name is:

Last name is:

Social security number is:

Gross sales are:

Commission rate is:

Earnings are:

* Use the Properties to update the grossSales to 5000.00
* Use the Properties to update the commissionRate to 0.1
* Use the ToString method to display all of the objects details
* Use the Property to display the earnings

Running the program should produce output similar to the following:

**Example Output:**

Employee information obtained by properties and methods:

First name is Sue

Last name is Jones

Social security number is 222-22-2222

Gross sales are $10,000.00

Commission rate is 0.06

Earnings are $600.00

Updated employee information obtained by ToString:

commission employee: Sue Jones social security number: 222-22-2222 gross sales: $5,000.00

commission rate: 0.10 earnings: $500.00

1. Referring to the UML diagram below, implement a BasePlusCommissionEmployee class (BasePlusCommissionEmployeec.cs). Descriptions of how to implement each method are also included below the UML diagram.

BasePlusCommissionEmployee

- firstName:string

- lastName:string

- socialSecurityNumber:string

- grossSales:decimal

- commissionRate:decimal

- baseSalary:decimal

+ <<property>> FirstName:string

+ <<property>> LastName:string

+ <<property>> SocialSecurityNumber:string

+ <<property>> GrossSales:decimal

+ <<property>> CommissionRate:decimal

+ <<property>> BaseSalary:decimal

+ BasePlusCommissionEmployee(first:string, last:string, ssn:string, sales:decimal, rate:decimal, salary:decimal)

+ Earnings():decimal

+ ToString():string

Methods for BasePlusCommissionEmployee class

BasePlusCommissionEmployees’s public services include a constructor, methods *Earnings()* and *ToString()*, and the public *properties* for manipulating the class’s instance variables. Each of its instance variables is private, and properties will be used to manipulate/access these variables.

GrossSales (set) Property and CommissionRate (set) Property will perform the same validation in their set method as described for the CommissionEmployee’s class.

BaseSalary (set) Property will throw an exception if the value being set is less than 0.

Method *Earnings*() – calculates a BasePlusCommissionEmployee’s earnings by multiplying the commissionRate by the grossSales then adding on the baseSalary and returning the result.

Method *ToString()* - one of the methods that every class inherits from class object, which is the root of the C# class hierarchy. It returns a string representing the object, and is called implicitly by an app whenever an object must be converted to a string representation, such as Console’s Write method. Use the format specifiers described in the CommissionEmployee’s class to format the details appropriately.

1. Create a test class named BasePlusCommissionEmployeeTest.cs. Inside the Main method, execute the following steps:

* Create a BasePlusCommissionEmployee object and invoke it’s constructor to initialise it with a set of values (you choose)
* Use the **Properties** to display each attributes value individually to the console and the earnings:

First name is:

Last name is:

Social security number is:

Gross sales are:

Commission rate is:

Base Salary is:

Earnings are:

* Use the Properties to update the baseSalary to 1000.00
* Use the ToString method to display all of the objects details
* Use the Property to display the earnings

Running the program should produce output similar to the following:

Employee information obtained by properties and methods:

First name is Bob

Last name is Lewis

Social security number is 333-33-3333

Gross sales are $5,000.00

Commission rate is 0.04

Earnings are $500.00

Base salary is $300.00

Updated employee information obtained by ToString:

base-salaried commission employee: Bob Lewis social security number: 333-33-3333

gross sales: $5,000.00 commission rate: 0.04 base salary: $1,000.00 earnings: $1,200.00