LAB DEMO Inheritance

# Objectives

* 1. Demonstrate how to declare a class that inherits from another.
  2. Demonstrate calling base class methods.
  3. Demonstrate how to declare and override virtual methods.
  4. Demonstrate polymorphism.
  5. Demonstrate differences between hiding and overriding.
  6. Demonstrate how to define virtual properties.

# Create a new project

* 1. Create a Console applications and call it “HRManager”.

# Inheritance Syntax

* 1. Add an Employee class:

public class Employee

{

public int EmpNum { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

}

* 1. Add an HourlyEmployee class:

public class HourlyEmployee:Employee

HourlyEmployee inherits from Employee

{

public decimal HourlyRate { get; set; }

public HourlyEmployee():base()

{

HourlyRate = 15.0M;

}

}

* 1. Add the following code to Main in program.cs:

static void Main(string[] args)

{

Employee emp = new Employee();

emp.EmpNum = 1;

emp.FirstName = "Steve";

emp.LastName = "Jobs";

Notice HourlyEmployee has all the properties of Employee plus HourlyRate.

HourlyEmployee hourEmp = new HourlyEmployee();

hourEmp.EmpNum = 2;

hourEmp.FirstName = "Bill";

hourEmp.LastName = "Gates";

hourEmp.HourlyRate = 15.00M;

}

# Overriding Base Class Methods

* 1. Add the following method to the Employee class:

public virtual string GetPaySummary()

{

return "No pay for base class employee.";

}

* 1. Add the following method to the HourlyEmployee class:

public override string GetPaySummary()

{

return "This employee is payed "+HourlyRate+" per hour";

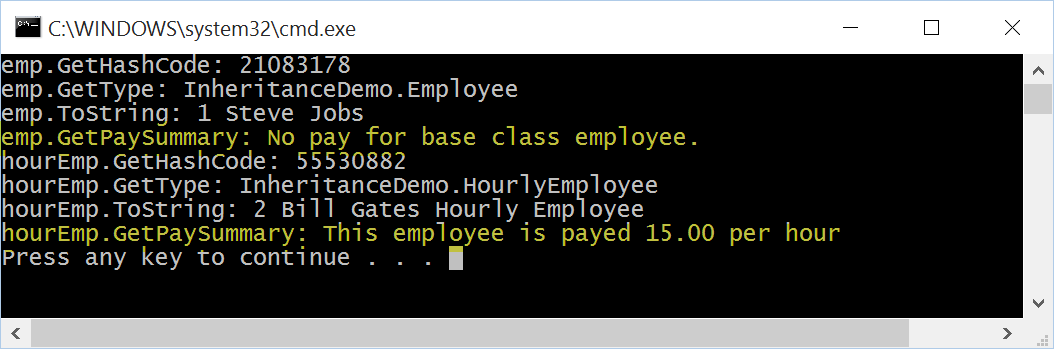
}

* + 1. Notice the override keyword.
  1. In Program.cs add the following two lines to Main():

Console.WriteLine("emp.GetPaySummary: " + emp.GetPaySummary());

Console.WriteLine("hourEmp.GetPaySummary: " + hourEmp.GetPaySummary());

* 1. Try it with Ctrl-F5:



* 1. Notice which method is called.

# Polymorphism

* 1. Change the HourlyEmployee code to:

HourlyEmployee hourEmp = new HourlyEmployee();

hourEmp.EmpNum = 2;

hourEmp.FirstName = "Bill";

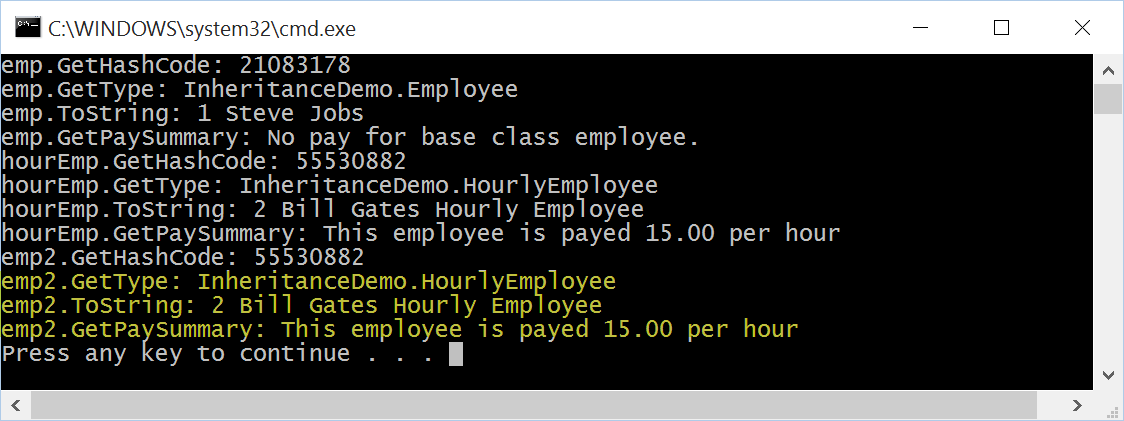
hourEmp.LastName = "Gates";

hourEmp.HourlyRate = 15.00M;

Console.WriteLine("hourEmp.GetPaySummary: " + hourEmp.GetPaySummary());

Employee emp2 = hourEmp;

* + 1. Console.WriteLine("emp2.GetPaySummary: " + emp2.GetPaySummary());Notice that we are assigning an HourlyEmployee object hourEmp into a variable declared as the base class Employee.
    2. Before running this code try to predict what emp2.ToString()); and emp2.GetPaySummary()); will display.
  1. Try it with Ctrl-F5:



* + 1. Notice that despite the fact that emp2 is an Employee class object, GetType(), ToString(), GetPaySummary() methods return the same value as the HourlyEmployee object!
    2. This is the effect of the override key word. When you override a method, the overridden version of the method is what will be called, no matter what base type variable you assign the object into.

# Method Hiding

* 1. In the Hourly Employee class remove the override keywords:

public ~~override~~ string GetPaySummary()

{

return "This employee is payed "+HourlyRate+" per hour";

}

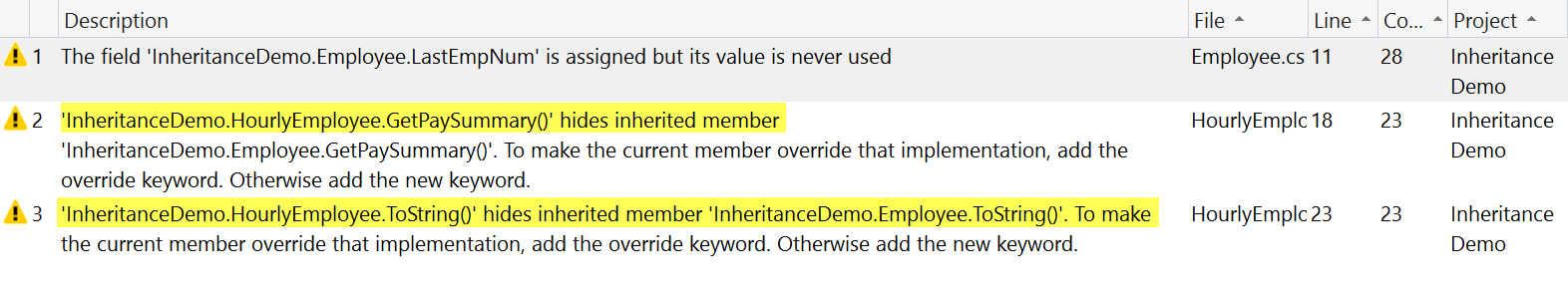
public ~~override~~ string ToString()

{

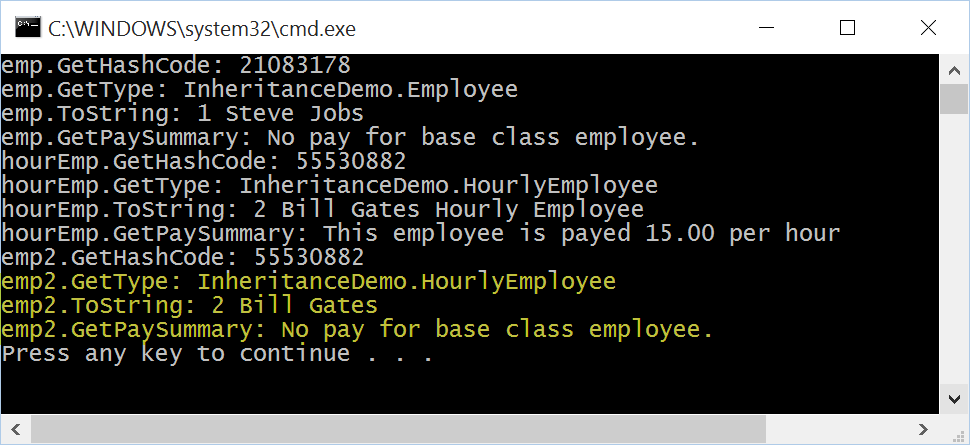
return base.ToString() + " Hourly Employee";

}

* 1. Build the project and notice the warning messages:



* 1. The warnings occur because we normally want to override class methods.
  2. Try it with Ctrl-F5:



* 1. Notice that emp2 now calls the ToString() and GetPaySummary() methods from the Employee class and not the HourlyEmployee class.
  2. Method hiding means that the method is only associated with an object when it is in a variable declared of the methods type.
  3. In this example, it creates an actual problem, our hourly employees pay information is lost if we do not override these methods! Method hiding is not always a wrong answer however. This is why you have the option to pick which approach you want to use.
  4. You can use the new keyword to suppress the warning:

new public string GetPaySummary()

{

return "This employee is payed "+HourlyRate+" per hour";

}

new public string ToString()

{

return base.ToString() + " Hourly Employee";

}

# Virtual Properties

* 1. Change GetPaySummary() into a property in Employee.cs:

public virtual string PaySummary

{

get { return "No pay for base class employee."; }

}

* 1. Change GetPaySummary() into a property in HourlyEmployee.cs as well:

public override string PaySummary

{

get { return "This employee is payed " + HourlyRate + " per hour"; }

}

* 1. In Main change the calls to the get methods into calls to the properties:

Console.WriteLine("emp.GetPaySummary: " + emp.PaySummary);

Console.WriteLine("hourEmp.GetPaySummary: " + hourEmp.PaySummary);

Console.WriteLine("emp2.GetPaySummary: " + emp2.PaySummary);

* 1. Try it with Ctrl-F5:

