C# : Classes and Objects

new World();

pluralsight
see what you can learn

Overview

- Classes versus Objects
- Constructors
- Inheritance
- Access modifiers
- Abstract classes
- Static classes
- Sealed classes
- Partial classes

Classes are to Objects as …

- **Classes define types**
  - State
  - Behavior
  - Access
- **Objects are instances of a type**
  - You can create multiple instances
  - Each instance holds different state
  - Each instance has same behavior

Constructors

- **Special methods to create objects**
  - Set default values
- **Multiple constructors allowed**
  - Overloaded methods must take different arguments

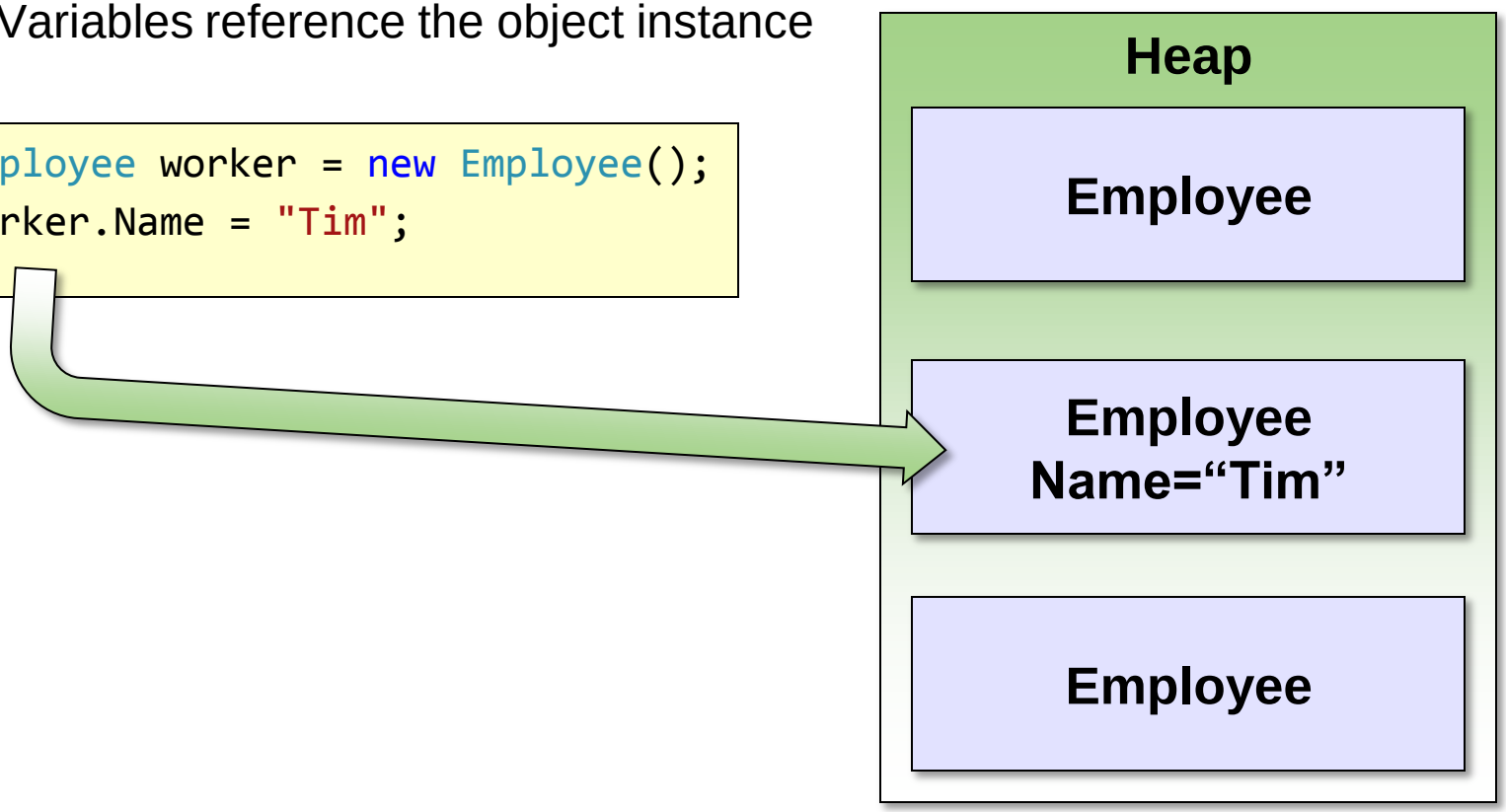
```
class Employee
{
    public Employee()
    {
        Name = "<empty>";
        Salaried = false;
    }

    // ...
```

Reference Types

- **Classes create reference types**
  - Object is stored on the “heap”
  - Variables reference the object instance

```
Employee worker = new Employee();
worker.Name = "Tim";
```



Object Oriented Programming

- C# is an OO language

Inheritance

Encapsulation

Polymorphism

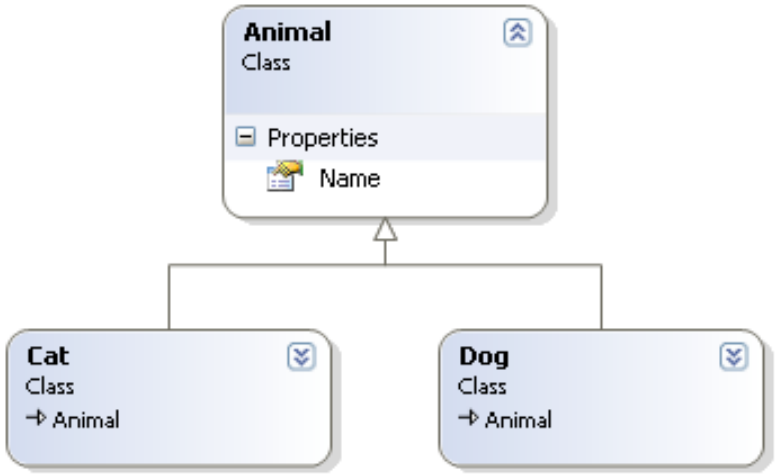
Inheritance

- **Create classes to extend other classes**
  - Classes inherit from System.Object by default
  - Gain all the state and behavior of the base class

```
class Animal
{
    public string Name { get; set; }
}

class Dog : Animal
{
}

class Cat : Animal
{
}
```



Access Modifiers

- **Keywords to declare the accessibility of a type or member**

Keyword	Applicable To	Meaning
public	Class, Member	No restrictions
protected	Member	Access limited to the class and derived classes
internal	Class, Member	Access limited to the current assembly
protected internal	Member	Access limited to current assembly and derived types
private	Member	Access limited to the class

Abstract

- **The abstract keyword**
  - Can apply to a class
  - Can also apply to members (methods, properties, indexers, events)
- **Abstract class cannot be instantiated**
  - Abstract class is designed as a base class
  - Must implement abstract members to make a concrete class

```
public abstract class Animal
{
    public abstract void PerformTrick();
}
```

```
public class Dog : Animal
{
    public override void PerformTrick()
    {
        // roll over
    }
}
```

Virtual

- **The virtual keyword creates a virtual member**
  - Can override the member in a derived class
  - Members are non-virtual by default
  - Virtual members dispatch on runtime type

```
public class Animal
{
    public virtual void PerformTrick()
    {
        // animal trick
    }
}
```

```
public class Dog : Animal
 {
     public override void PerformTrick()
     {
         // perform dog trick
         //  and then animal trick
         base.PerformTrick();
     }
 }
```

Static

- **Static members are members of the type**
  - Cannot invoke the member through an object instance
- **Static classes can have only static members**
  - Cannot instantiate a static class

```
public double Circumference
{
    get { return Diameter * Math.PI; }
}

public double Diameter
{
    get; set;
}
```

Sealed

- **Sealed classes cannot be inherited**
  - Prevent extensibility or misuse
  - Some framework classes sealed for performance and security implications

```
public class MyString : System.String
{                                // error!

}
```

Partial classes

- **Partial classes frequently generated by VS designer**
- **Partial class definitions can span multiple files**
  - But only in the same project
- **Partial method definitions are extensibility points**
  - Optimized away if no implementation provided

```
public partial class Animal
{
    public string Name { get; set; }
    partial void OnNameChanged();
}
```

```
public partial class Animal
{
    partial void OnNameChanged()
    {
        // ....
    }
}
```

Summary

- **C# gives you everything you need for OOP**
  - Encapsulation
  - Inheritance
  - Polymorphism
- **Additional features for performance, convenience, extensibility**
  - Static classes
  - Sealed classes
  - Partial classes