شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

C# Object Oriented Programming

Tahaluf Training Center 2021

day 2

1. **What Is Inheritance?**
2. Calling Base Class Constructors and Members
3. Enums

What is Inheritance?

Inheritance is one of the fundamental attributes of object-oriented programming.

It allows you to define a child class that reuses (inherits), extends, or modifies the behavior of a parent class.

The class whose members are inherited is called the base class.

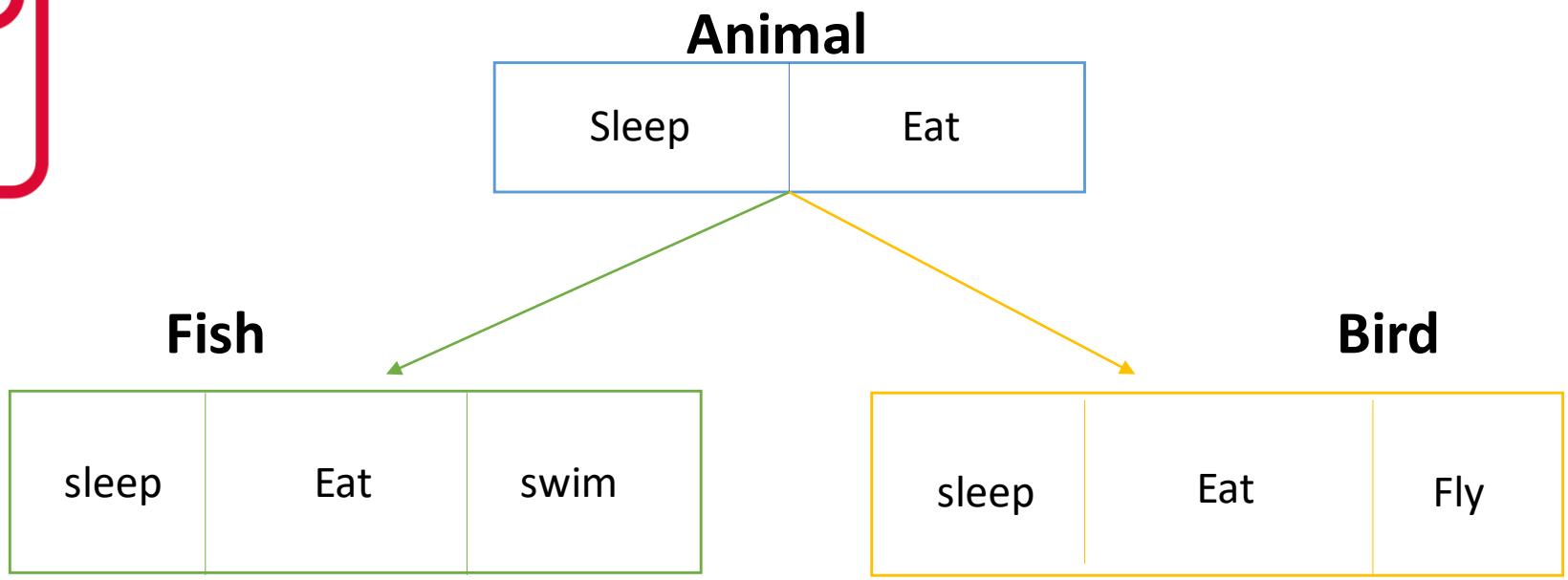
What is Inheritance?

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

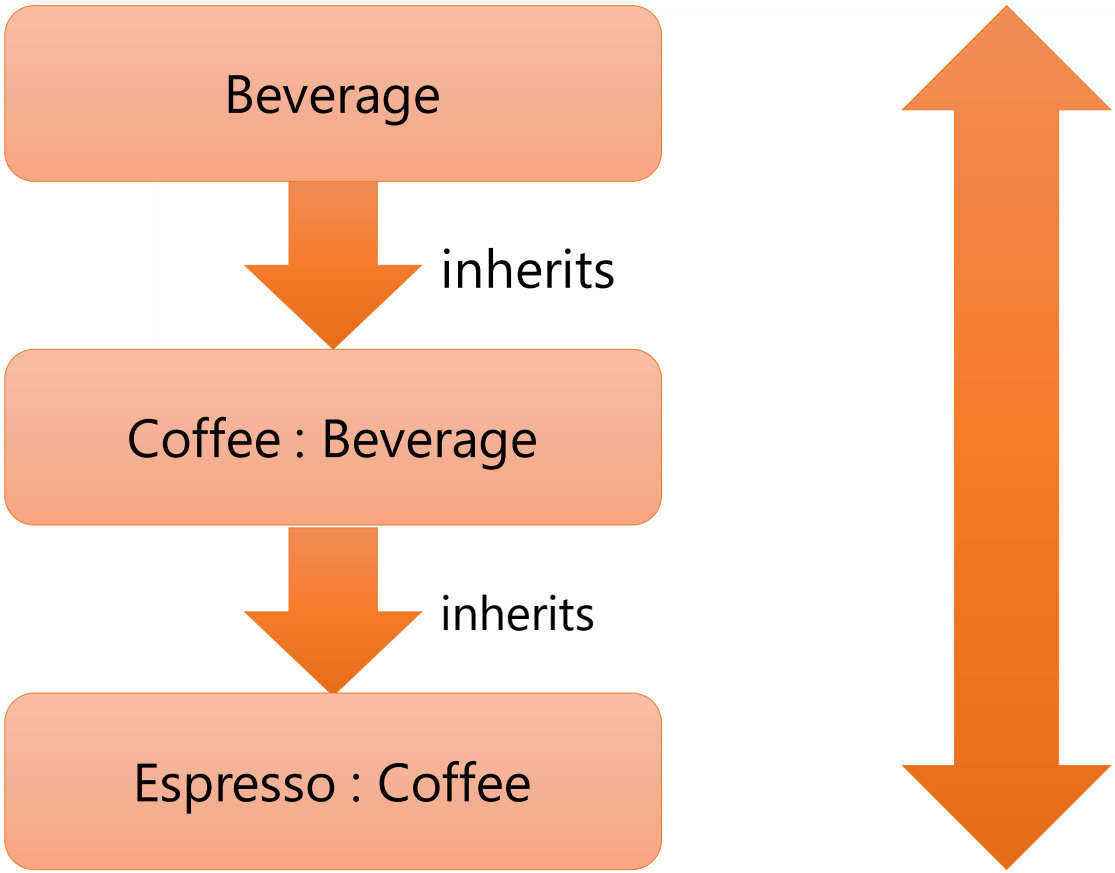
The class that inherits the members of the base class is called the derived class.

C# and .NET support single inheritance only,
How can we solve this problem?

What is Inheritance?



What is Inheritance?



```
public class Beverage
    {
        private int ServingTemperature1;
        protected int ServingTemperature2;
        protected internal int ServingTemperature3;
        internal int ServingTemperature4;
        public int ServingTemperature5;
    }
    public class Coffee : Beverage
    {
        public Coffee()
        {
            // what variables we can access ?
        }

    }
```

Access Modifiers

- Access modifiers are an integral part of object-oriented programming. Access modifiers are used to implement encapsulation of OOP. Access modifiers allow you to define who does or who doesn't have access to certain features.

Access Modifiers

1. **public modifier**:
- Can be accessed by objects of the class
- Can be accessed by derived classes

2. **private modifier**:
- Cannot be accessed by object
- Cannot be accessed by derived classes

3.**protected modifier**: A protected member is accessible from within the class in which it is declared, and from within any class derived from the class that declared this member.

- Cannot be accessed by object
- Can be accessed By derived classes

4. **internal modifier** :

The internal keyword is an access modifier for types and type members.

In same assembly (public)

- Can be accessed by objects of the class
- Can be accessed by derived classes

In other assembly (internal)

- Cannot be accessed by object
- Cannot be accessed by derived classes

protected internal modifier: protected internal member is accessible from any class in the same assembly, including derived classes.

The protected internal access modifier seems to be a confusing but is a union of protected and internal in terms of providing access but not restricting.

protected internal modifier allows:

- Inherited types, even though they belong to a different assembly, have access to the protected internal members.

- Types that reside in the same assembly, even if they are not derived from the type, also have access to the protected internal members.

Creating Abstract Class

Data **abstraction** is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces (which you will learn more about in the next chapter).

Creating Abstract Class

The abstract keyword is used for classes and methods:

1. **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

2. **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

Creating Abstract Class


```
abstract class Animal
        {
            // Abstract method (does not have a body)
            public abstract void animalSound();
            // Regular method
            public void sleep()
            {
                Console.WriteLine("Zzz");
            }
        }

        // Derived class (inherit from Animal)
        class Bee : Animal
        {
            public override void animalSound()
            {
            // The body of animalSound() is provided here
                Console.WriteLine("The Bee says: buzz");
            }
        }
```

Create an Animal class with one method for making sound. Then create a Dog and a Cat (which derive from Animal) and make them bark or meow respectively.

Creating Sealed Class

If you don't want other classes to inherit from a class, use the sealed keyword.

```
sealed class Vehicle
{

}

class Car : Vehicle
{

}
```

Virtual Keyword

Use the virtual keyword to create members that you can override in derived classes

```
 public virtual void animalSound()
{

Console.WriteLine("nothing");

}
```

Virtual Keyword

```
abstract class Animal
        {
            public virtual void animalSound() {
                Console.WriteLine("nothing");

            }
            public virtual void sleep()
            {
                Console.WriteLine("Zzz");
            }
        }

        class Bee : Animal
        {
            public override void animalSound()
            {
                Console.WriteLine("The Bee says: buzz");
            }

        }

        class Dog : Animal
        {
            public override void animalSound()
            {
                Console.WriteLine("The dog says: how how");
            }
        }
```

day 2

1. What Is Inheritance?
2. **Calling Base Class Constructors and Members**
3. Enums

Calling Base Class Constructors and Members

In c#, the base keyword is useful to access base class members such as properties, methods, etc. in the derived class.

By using base keyword, we can call a base class method that has been overridden by another method in the derived class .

Calling Base Class Constructors and Members


```
public class A // This is the base class.
   {
      public A(int z)
      {
         // Executes some code in the constructor.
         Console.WriteLine("Base constructor A()");
      }
   }

   public class B : A // This class derives from the previous
class.
   {
      public B(int value)
      : base(value)
      {
         // The base constructor is called first.
         // ... Then this code is executed.
         Console.WriteLine("Derived constructor B()");
      }
   }
```

day 2

1. What Is Inheritance?
2. Calling Base Class Constructors and Members
3. **Enums**

Enum Data Type

An enum is a special "class" that represents a group of constants (unchangeable/read-only variables).

To create an enum, use the enum keyword (instead of class or interface), and separate the enum items with a comma:

```
enum Level { Low, Medium, High }
```

You can access enum items with the dot syntax:

```
class Program
    {
        enum Level
        {
            Low,
            Medium,
            High
        }
        static void Main(string[] args)
        {
            Level myVar = Level.Medium;
            Console.WriteLine(myVar);
        }
    }
```

Enum Data Type

```
enum Level
{       Low,        Medium,        High    }

static void Main(string[] args)
{
    Level myVar = Level.Medium;
    switch (myVar)
    {
        case Level.Low:
            Console.WriteLine("Low level");
            break;
        case Level.Medium:
            Console.WriteLine("Medium level");
            break;
        case Level.High:
            Console.WriteLine("High level");
            break;
    }
}
```

Create enum with name **month** and its data members are the name of months like **Jan**, **Feb**, **mar**, **Apr**, **may**. print the default integer values of these enums. An explicit cast is required to convert from enum type to an integral type.

Enum Data Type


```
enum Months
        {
            January,    // 0
            February,   // 1
            March,      // 2
            April,      // 3
            May,        // 4
            June,       // 5
            July        // 6
        }

        static void Main(string[] args)
        {
            int myNum = (int)Months.April;
            Console.WriteLine(myNum);
        }
```