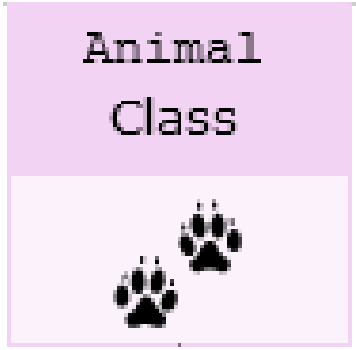
Session: 7

Inheritance and Polymorphism

Objectives

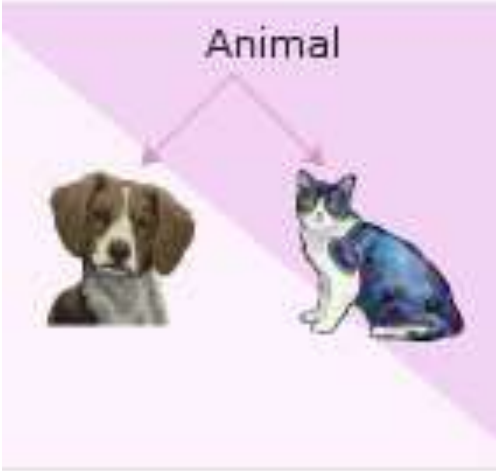
- Define and describe inheritance
- Explain method overriding
- Define and describe sealed classes
- Explain polymorphism

- The purpose of inheritance is to reuse common methods and attributes among classes without recreating them.
- Reusability enables to use the same code in different applications with little or no changes.



Example:

- Consider a class named **`Animal`** which defines attributes and behavior for animals.
- If a new class named **`Cat`** has to be created, it can be done based on **`Animal`**.



Implementing Inheritance

- To derive a class from another class, insert a colon after the name of the derived class, followed by the name of the base class.
- The derived class can now inherit all non-private methods and attributes of the base class.
- The following syntax is used to inherit a class in C#:

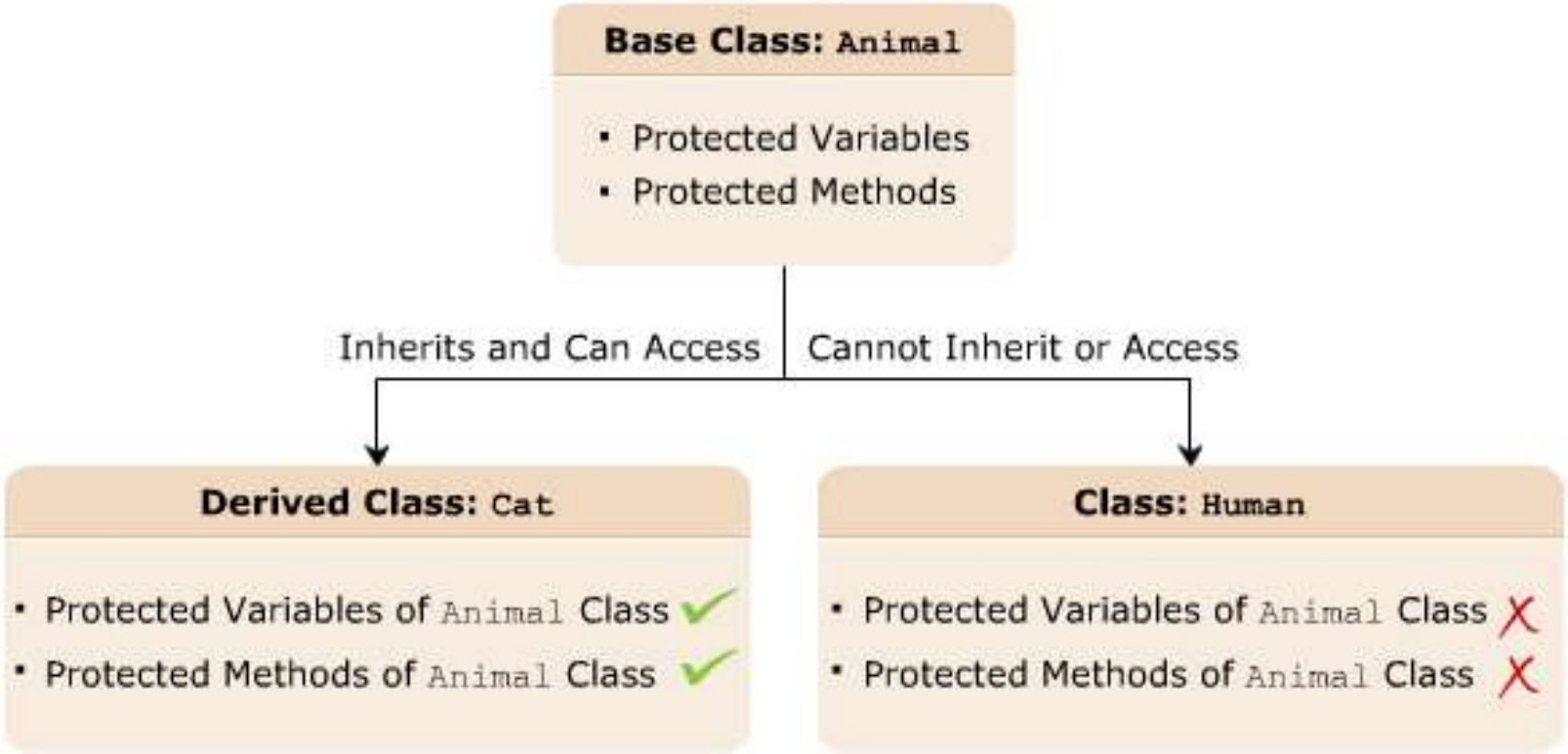
```
<DerivedClassName> : <BaseClassName>
```

where,

- **DerivedClassName:** Is the name of the newly created child class.
- **BaseClassName**: Is the name of the parent class

protected Access Modifier

- used to protect the members in a class so that they are accessed only by the class they declared or by the child class.



- The following syntax shows the use of the **base** keyword:

```
class <Parent>{
    <accessmodifier> <return type> <MethodA>(){…}
}

class <Child>:<Parent> {
    <accessmodifier> <return type> <MethodA>(){
        ...
        base.<MethodA>();
    }
}
```

```
class Animal
{
    public void Eat(){}
}
class Dog : Animal
{
    public void Eat(){}
    public static Main(string args[])
    {
        Dog objDog = new Dog();
        objDog.Eat;
        base.Eat();
    }
}
```

- Used as a modifier in order to hide the base class method :

```
class <Parent> {
    <access modifier> <returntype> <BaseMethod>(){}
}

class <Child>:<Parent> {
      new <access modifier> <returntype> <BaseMethod>(){}
}
```

- Used as an operator to creates an object :

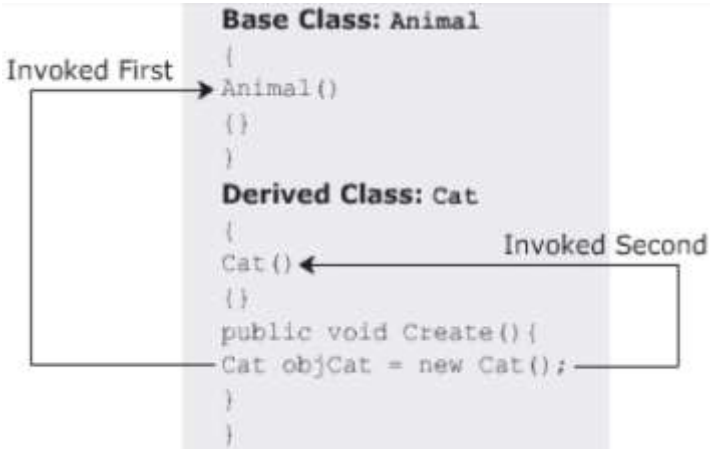
```
Employees objEmp = new Employees();
```

- In C#:

the base class constructor call be invoked by either instantiating the derived class or the base class
the constructor of base class goes first, and then constructor of derived class.

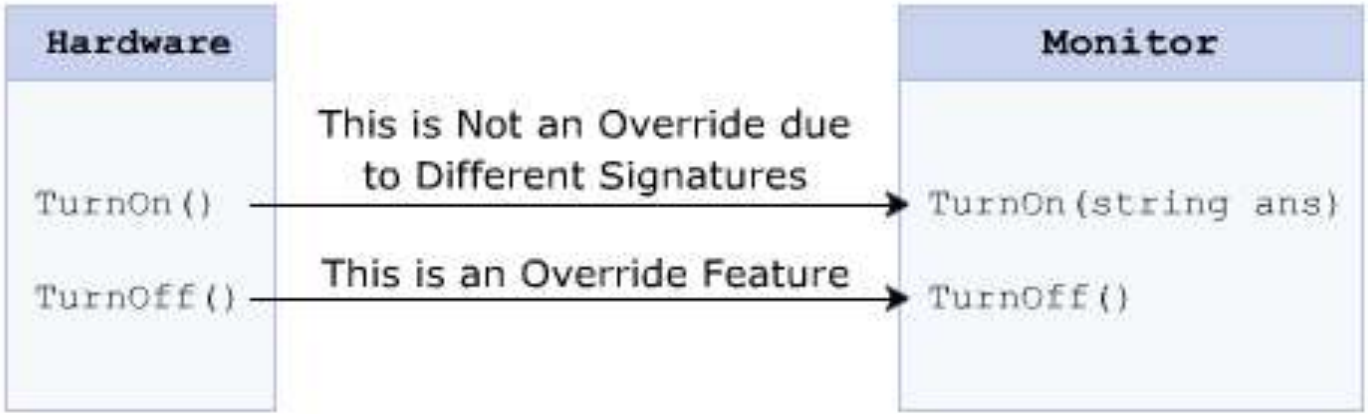
the base class constructor can be invoked by using base keyword in the derived class constructor declaration.

- However, constructors cannot be inherited



Method Overriding

- Allows a method in derived class having the same name and signature as the method in the base class, to be rewritten to redefine new behavior.
- Ensures reusability while inheriting classes.
- Is implemented in the derived class , is known as the Overridden Base Method.
- The following figure depicts method overriding:



virtual and override Keywords

- To implement the **overriding** feature, using the **virtual** and **override** keywords:

```
<access_modifier> virtual <return_type> <MethodName>(<parameters);
```

```
<access_modifier> override <return_type> <MethodName>(<parameters);
```

where,

- **virtual: Is a** keyword used to declare a method in the base class that can be overridden by the derived class.
- **override:** Is a keyword used to declare a method in the derived class

- A sealed class is a class that prevents inheritance.
- The features :

be declared by preceding the class keyword with the **sealed** keyword.

cannot be a base class

Syntax

```
sealed class <ClassName>
{
        //body of the class
}
```

Sealed Methods

prevents the method from further overriding by derived class.

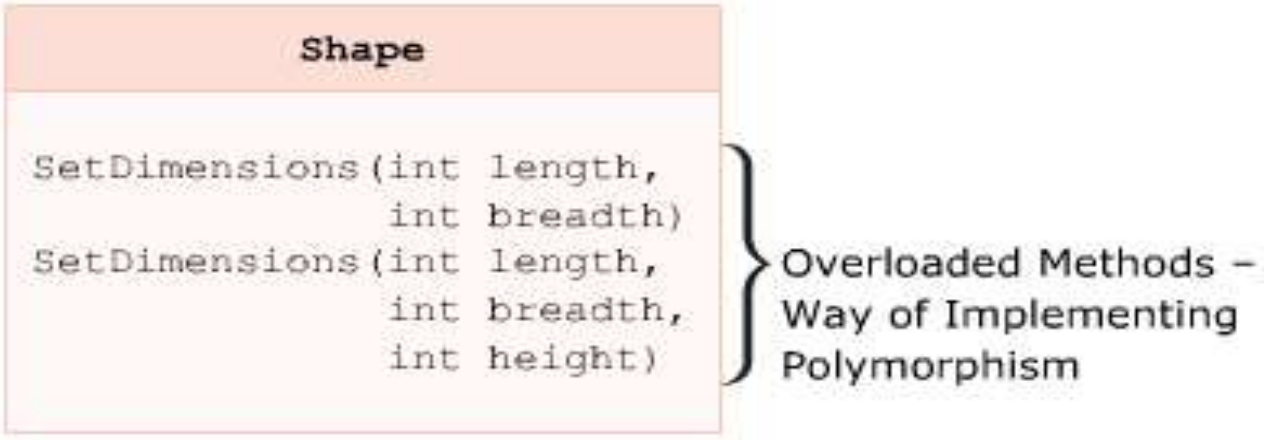
When the derived class overrides a base class method, variable, property or event, then the new method, variable, property, or event can be declared as sealed.

- The following syntax is used to declare an overridden method as sealed:

```
sealed override <return_type> <MethodName>() {

}
```

- Polymorphism is the ability of an entity to behave differently in different situations.
- Polymorphism is derived from two Greek words, namely **Poly** and **Morphos**, meaning forms. Polymorphism means existing in multiple forms.
- Polymorphism allows methods to function differently based on the parameters and their data types. For example:



Summary

- Inheritance allows to create a new class from another class, thereby inheriting its common properties and methods.
- Inheritance can be implemented by writing the derived class name followed by a colon and the name of the base class.
- Method overriding is a process of redefining the base class methods in the derived class.
- Methods can be overridden by using a combination of virtual and override keywords within the base and derived classes respectively.
- Sealed classes are classes that cannot be inherited by other classes, declared by using the sealed keyword.
- Polymorphism is the ability of an entity to exist in two forms that are compile-time polymorphism and run-time polymorphism.