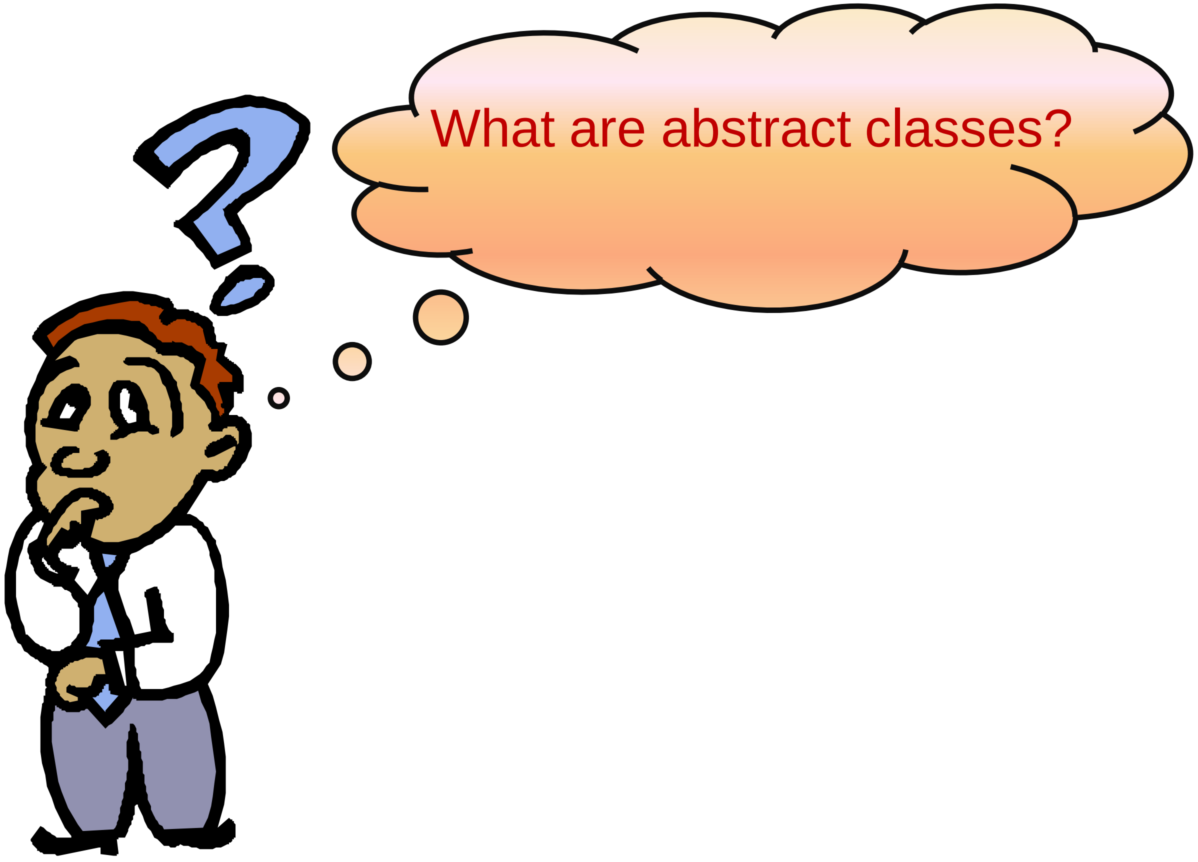
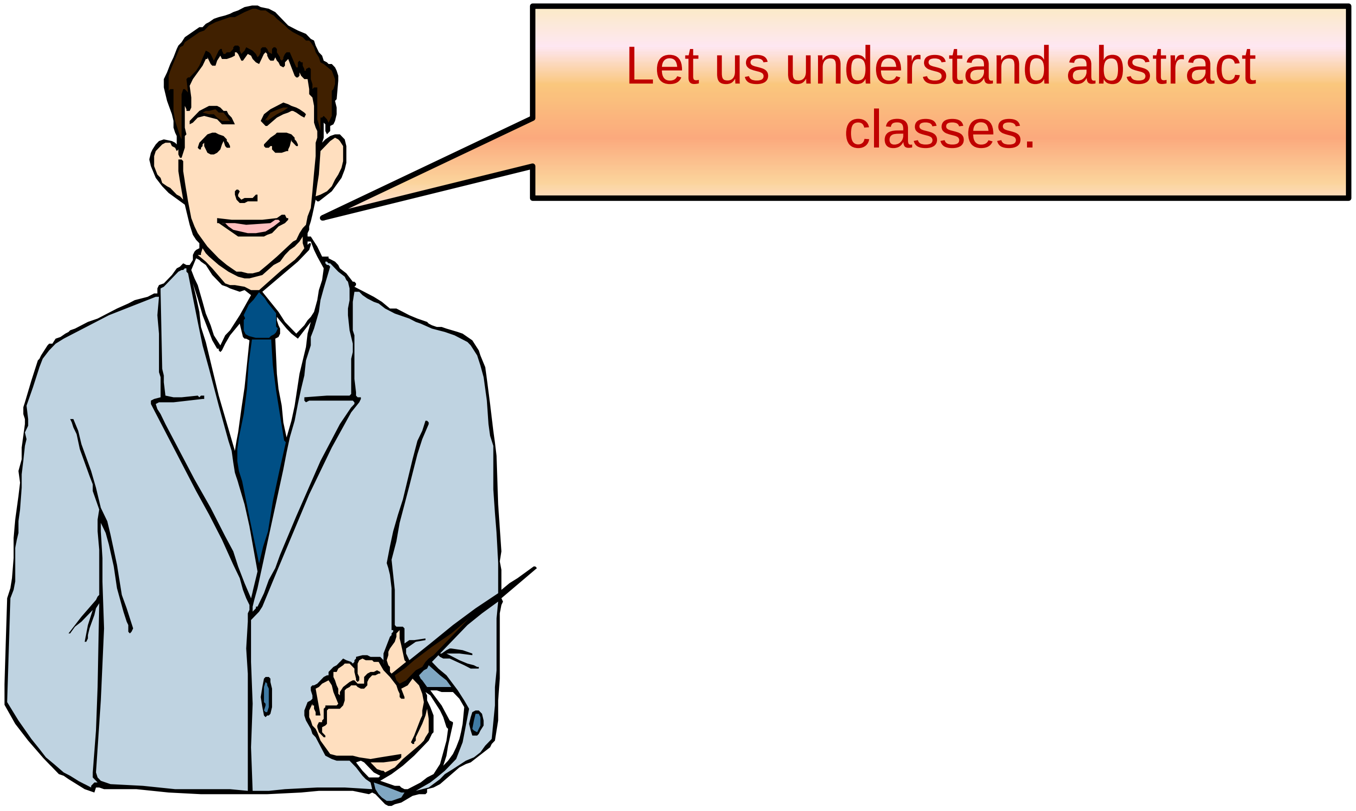
- In this session, you will learn to:
  - Use abstract classes


What are abstract classes?


Let us understand abstract classes.

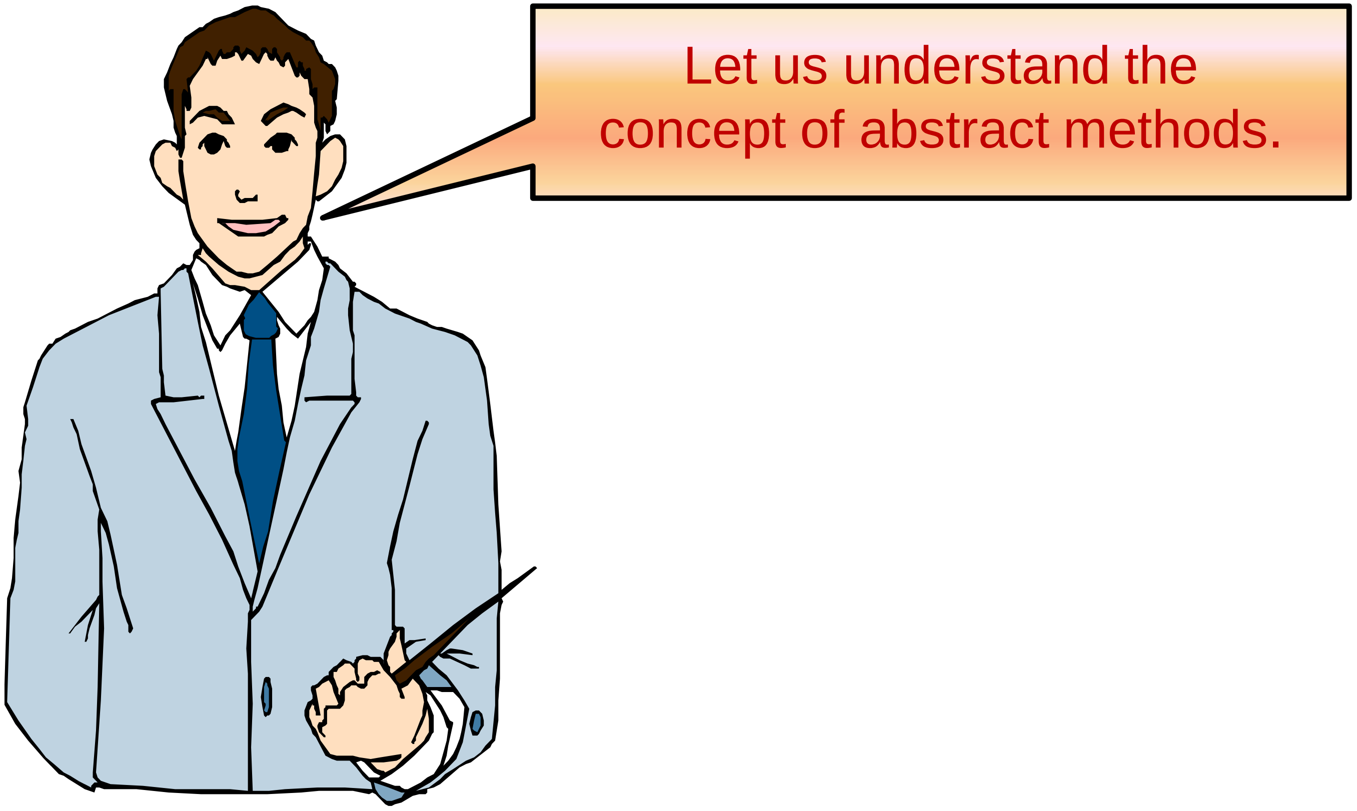
- In C#, the abstract class implies that the class is incomplete and cannot be used directly. To use an abstract class, other classes can be derived from it.
- Abstract classes contain abstract methods, which can be implemented by the derived class.
- You cannot create instance of an abstract class to use its member functions.
- You can override the methods of the abstract class in the base class and provide a new functionality to it.
- Polymorphism can be implemented by using abstract classes and virtual functions.

- The following code snippet shows the declaration of an abstract class:

```
abstract class ExampleAbstract /*This is an abstract
                                                       class*/
{
        …………
}
Class TestAbstract
{
   staic void Main(string[] args)
   {
     ExampleAbstract obj= new ExampleAbstract();
     //An abstract class cannot be instantiated
   }
}
```

- The following rules govern the use of an abstract class:
  - You cannot create an instance of an abstract class.
  - You cannot declare an abstract method outside an abstract class.
  - You cannot declare an abstract class as sealed.
  - A class, which is derived from an abstract class, must override all the methods of the abstract class.
  - If a derived class does not implement all of the abstract methods in the base class, then the derived class must also be specified as abstract.

What are abstract methods?


Let us understand the concept of abstract methods.

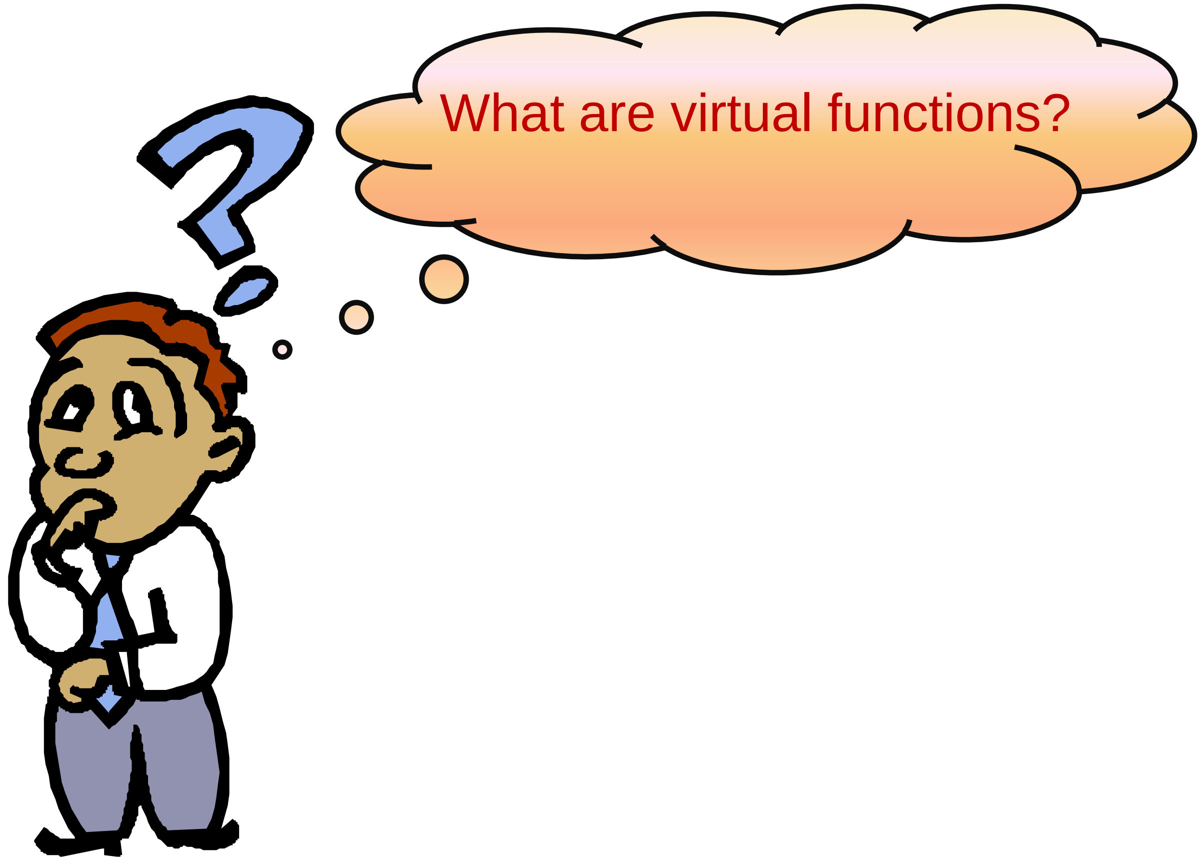
- Abstract methods are methods without any body.
- The implementation of an abstract method is done by the derived class.
- When a derived class inherits the abstract method from the abstract class, it must override the abstract method to provide it a new functionality.
- This requirement is enforced at compile time, and is also called dynamic polymorphism.
- The syntax for using the abstract method is:

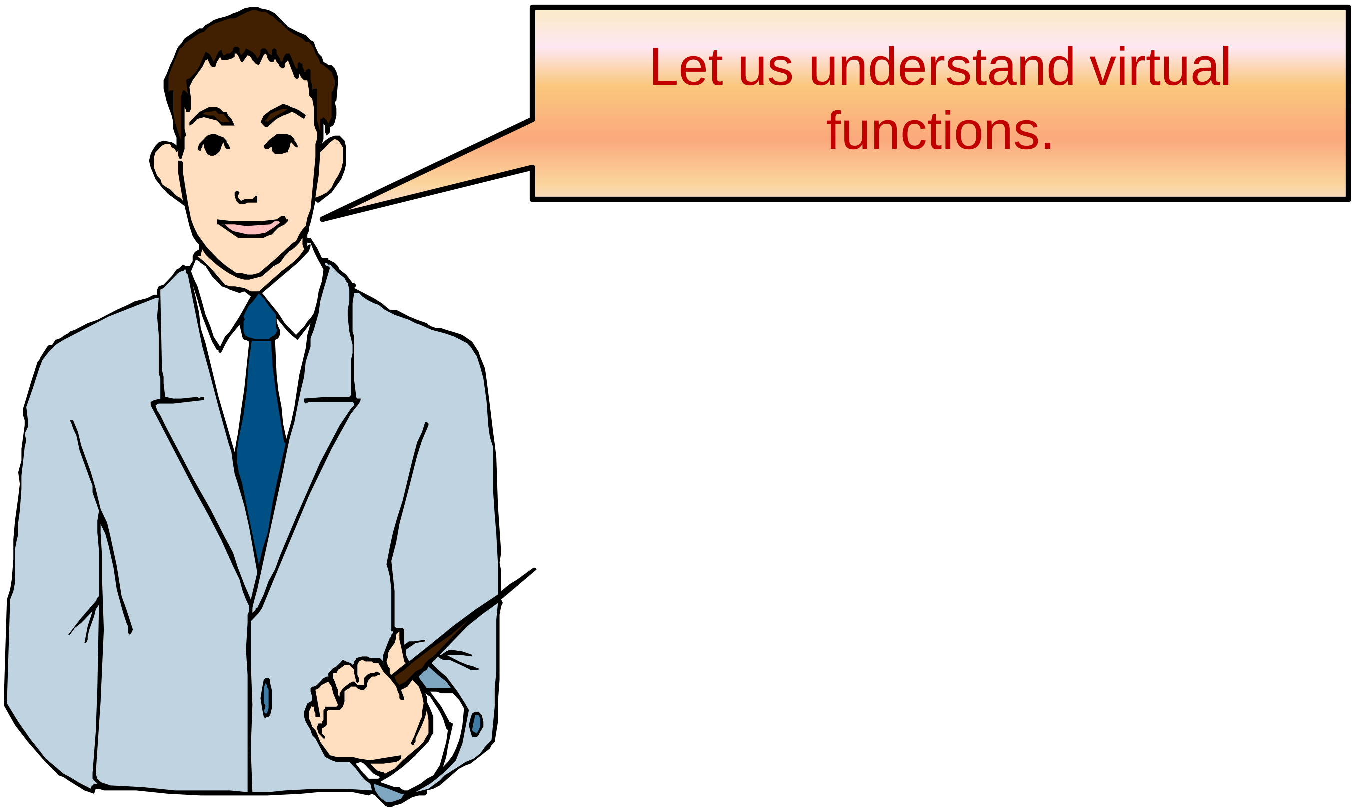
```
[access-modifiers] abstract return-type
 method-name ([parameters]);
```

- The following code shows the use of abstract classes with abstract methods:

```
using System;
abstract class Animal
{
    public abstract void FoodHabits();
}
class Carnivorous: Animal
{
    public override void FoodHabits(     )
    {
     Console.WriteLine("The Carnivorous animals eat only
                             meat");
    }
 }
```

```
class Herbivorous: Animal
{
   public override void FoodHabits()
   {
            Console.WriteLine("The Herbivorous animals
                                 eat only plants");
    }
 }
class Implement
{
   public static void Main()
   {
            Carnivorous cn=new Carnivorous();
            Herbivorous hb=new Herbivorous ();
            cn.FoodHabits();
            hb.FoodHabits();
 }}
```


What are virtual functions?


Let us understand virtual functions.

- When you have a function defined in a class that you want to be implemented by the inherited classes, you can use a virtual function.
- The functionality of the virtual function is modified by the inherited class according to its requirement and the call to the method is decided at runtime.
- To declare a virtual function, the `virtual` keyword is used before the return type of the function and after the access modifier of the function.

- The following code shows the use of virtual functions:

```
using System;
namespace VirtualFunction
{
 class Animal
 {
      public virtual void FoodHabits()
      {
            Console.WriteLine("Animals has different
                           food habits");
      }
 }
```

```
class Carnivorous : Animal
 {
     public override void FoodHabits()
     {
       Console.WriteLine("The Carnivorous animals
                                 eat only meat");
     }
 }
class Herbivorous : Animal
 {
     public override void FoodHabits()
     {
```

```
 Console.WriteLine("The Herbivourous animals eat only
                 plants");
    }
}
class Implement
{
    public void callFunction(Animal Cr)
    {
        Cr.FoodHabits();
    }
 }
```

```
class ClassMain
 {
      public static void Main()
      {
             Implement Imp = new Implement();
             Carnivorous cn = new Carnivorous();
             Herbivorous hb = new Herbivorous();
             Imp.callFunction(cn);
             Imp.callFunction(hb);
             Console.ReadLine();
      }
   }
}
```

- In this session, you learned that:
  - An abstract class is an incomplete class that cannot be used directly.
  - To use an abstract class, other classes can derive from it.