- it gains the features of the class that inherits from
- Parent(base class) - class that's being inherited from
- Child (derived class) - class that is inheriting
- items that exists in the parent class are also available to the child class

Access modifiers:

- private : available to base class only
- protected : available to base and child only
- public : available to everyone

```
using UnityEngine;
using System.Collections;

//This is the base class which is
//also known as the Parent class.
public class Fruit
{
    public string color;

    //This is the first constructor for the Fruit class
    //and is not inherited by any derived classes.
    public Fruit()
    {
        color = "orange";
        Debug.Log("1st Fruit Constructor Called");
    }

    //This is the second constructor for the Fruit class
    //and is not inherited by any derived classes.
    public Fruit(string newColor)
    {
        color = newColor;
        Debug.Log("2nd Fruit Constructor Called");
    }

    public void Chop()
    {
        Debug.Log("The " + color + " fruit has been chopped.");
    }
    public void SayHello()
    {
        Debug.Log("Hello, I am a fruit.");
    }
}


using UnityEngine;
using System.Collections;

//This is the derived class which is
//also known as the Child class.
public class Apple : Fruit
{
    //This is the first constructor for the Apple class.
    //It calls the parent constructor immediately, even
    //before it runs.
    public Apple()
    {
        //Notice how Apple has access to the public variable
        //color, which is a part of the parent Fruit class.
        color = "red";
        Debug.Log("1st Apple Constructor Called");
    }

    //This is the second constructor for the Apple class.
    //It specifies which parent constructor will be called
    //using the "base" keyword.
    public Apple(string newColor) : base(newColor)
    {
        //Notice how this constructor doesn't set the color
        //since the base constructor sets the color that
        //is passed as an argument.
        Debug.Log("2nd Apple Constructor Called");
    }
}
```

```
using UnityEngine;
using System.Collections;

public class FruitSalad : MonoBehaviour
{
    void Start ()
    {
        //Let's illustrate inheritance with the
        //default constructors.
        Debug.Log("Creating the fruit");
        Fruit myFruit = new Fruit();
        Debug.Log("Creating the apple");
        Apple myApple = new Apple();

        //Call the methods of the Fruit class.
        myFruit.SayHello();
        myFruit.Chop();

        //Call the methods of the Apple class.
        //Notice how class Apple has access to all
        //of the public methods of class Fruit.
        myApple.SayHello();
        myApple.Chop();

        //Now let's illustrate inheritance with the
        //constructors that read in a string.
        Debug.Log("Creating the fruit");
        myFruit = new Fruit("yellow");
        Debug.Log("Creating the apple");
        myApple = new Apple("green");

        //Call the methods of the Fruit class.
        myFruit.SayHello();
        myFruit.Chop();

        //Call the methods of the Apple class.
        //Notice how class Apple has access to all
        //of the public methods of class Fruit.
        myApple.SayHello();
        myApple.Chop();
    }
}
```