- is the practice of changing the parent class method in a child class
- we use the word “virtual” in the parent class
- we use the word “override” in the child class
- we can keep functionality of the base class by using **base.methodName();** and just extra functionality in the child class

```
using UnityEngine;
using System.Collections;

public class Fruit
{
    public Fruit ()
    {
        Debug.Log("1st Fruit Constructor Called");
    }

    //These methods are virtual and thus can be overridden
    //in child classes
    public virtual void Chop ()
    {
        Debug.Log("The fruit has been chopped.");
    }

    public virtual void SayHello ()
    {
        Debug.Log("Hello, I am a fruit.");
    }
}
```

```
using UnityEngine;
using System.Collections;

public class Apple : Fruit
{
    public Apple ()
    {
        Debug.Log("1st Apple Constructor Called");
    }
```

```
    //These methods are overrides and therefore
    //can override any virtual methods in the parent
    //class.
    public override void Chop ()
    {
        base.Chop();
        Debug.Log("The apple has been chopped.");
    }

    public override void SayHello ()
    {
        base.SayHello();
        Debug.Log("Hello, I am an apple.");
    }
}


using UnityEngine;
using System.Collections;

public class FruitSalad : MonoBehaviour
{
    void Start ()
    {
        Apple myApple = new Apple();

        //Notice that the Apple version of the methods
        //override the fruit versions. Also notice that
        //since the Apple versions call the Fruit version with
        //the "base" keyword, both are called.
        myApple.SayHello();
        myApple.Chop();

        //Overriding is also useful in a polymorphic situation.
        //Since the methods of the Fruit class are "virtual" and
        //the methods of the Apple class are "override", when we
        //upcast an Apple into a Fruit, the Apple version of the
        //Methods are used.
```

```
        Fruit myFruit = new Apple();
        myFruit.SayHello();
        myFruit.Chop();
    }
}
```