FRUIT INHERITANCE

Create Parent Child relationships by setting up your classes:

Create three new non-static classes and make them all public. (Parent is called Fruit, Child is Apple, and another Child is Banana)

The first thing you need to know is how to create parent base class and child subclass relationship.

Note: In this case, `Apple` is considered a Child of type `Fruit`. **Please pay attention to that there is a colon in between.**

The second thing you need to know is how to set/get properties, there is a long version and a short version. The short version is considered "auto implemented" because it takes on the value that the class is passing. The long version is specific to what your calling to take on in that class. We are only concerned with using the short version for this project.

Note: This is the way you will be creating your two child classes.

Short version:
Prop + tab tab //Type "prop" then press the tab twice.

```
public int MyProperty { get; set; }
```

Long Version:
propfull + tab tab // Type "propfull" then press the tab twice.

```
private int myVar;
        public int MyProperty
        {
                get { return myVar;}
                set { myVar = value;}
        }
```

Note: Our class has a parent child relationship below with the "Apple" as of type "Fruit":

```
public class Apple : Fruit
 {Note: We used a short version of get set property for the variable string variable attribute Color:

    // Get and Set the Property Color.
    public string Color { get; set; }
```

The next thing you need to know is about the constructor (below). A constructor can be empty, called a "default" constructor. It helps to represent the instantiation of a class directly by placing the default constructor. In this way, we control the initialization of the class and don't allow anyone else to initialize the class.

```
        //Default Apple Constructor.
        public Apple()
        {
        }
```

Next we create a method that passes a string and an integer.
Note: The capitalization of Color and Weight to pass the string and integer.

```
        // Call's the Apple Method.
        public Apple(string color, int weight)
        {
            Color = color;
            Weight = weight;
        }

    }
```

Your parent class will look like below:

```
    public class Fruit
    {
        public int Weight { get; set; }
    }
```

/*========================== On the Console ========================*/

Instantiating Objects:

An object can be instantiated in one of two ways (first is the long way and then there is the simplified way).

```
            // Note: The constructors in the subclasses are getting weight from the Fruit class,
                     then passing properties to the parameter values of the objects.

            // name and create a new variable of type Apple1, then pass the property values.
            Apple Apple1;
            Apple1 = new Apple("Red", 5); //instantiate an object instance of a new object, call constructor,
new returns reference to object.
```

Simplified: `Apple Apple1 = new Apple("Red", 5); //We name the variable type in the beginning and push it.`

Note: This object will pass a string **"Red"** and a number **5.**

Next we print out the object(s) by calling the method PrintApple() and placing the object into the parenthesis: (Print is a keyword, apple is an object type)

PrintApple(Apple1);

Now you can use a method to print out the apple like this below:

Note: Your basically calling the Object Apple referencing as applePassedIn. It doesn't care what you name that word because it is just a reference to the object Apple. Whatever the attribute being passed in will show regardless. In this case, the attribute color is **"Red"** and the number is **5**. Attributes are already defined.

```
        private static void PrintApple(Apple applePassedIn)
        {
            Console.Write("\nThis Apple weighs:  {0} oz, and its Color is: {1}", applePassedIn.Weight,
                                                                                  applePassedIn.Color+".");
        }
```