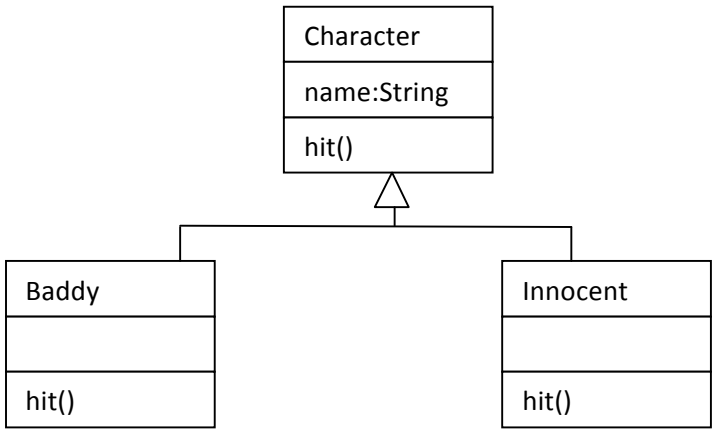
C# Polymorphism

Polymorphism refers to how classes and subclasses behave when they possess the same methods.

Overriding describes a sub class having the same named method as a superclass.



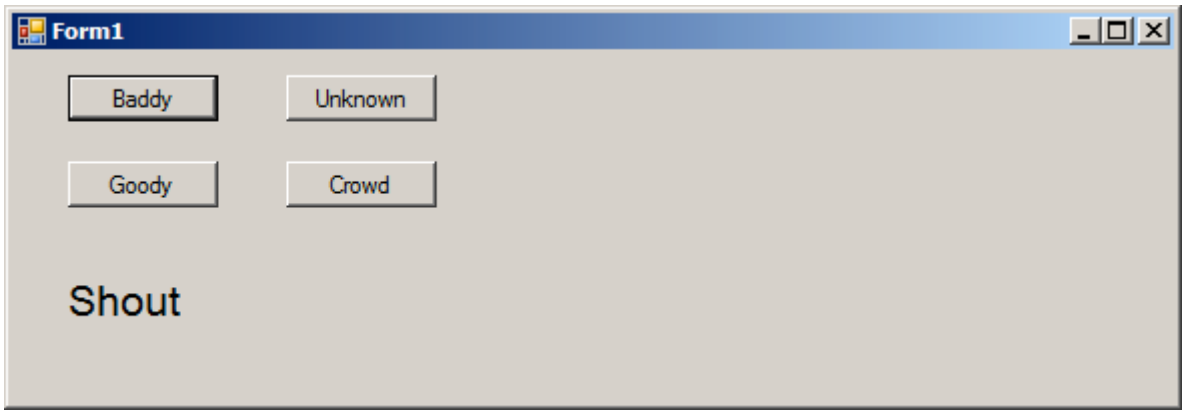
This is the basic Character class

```
class Character
{
    protected String name;
    public Character()
    {
        name = "nobody";
    }
    public String hit()
    {
        return "nothing";
    }
}
```

The other classes have their own constructors (assigning a new value to name) and hit methods (returning some new message).

Both need to be in the same namespace and to inherit from Character

```
class Baddy : Character
```



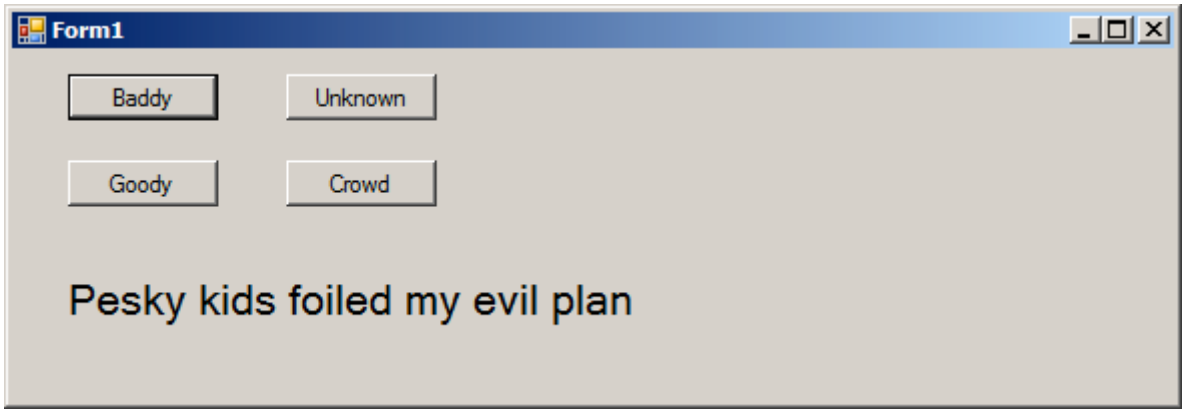
The front end form creates an instance of each class:

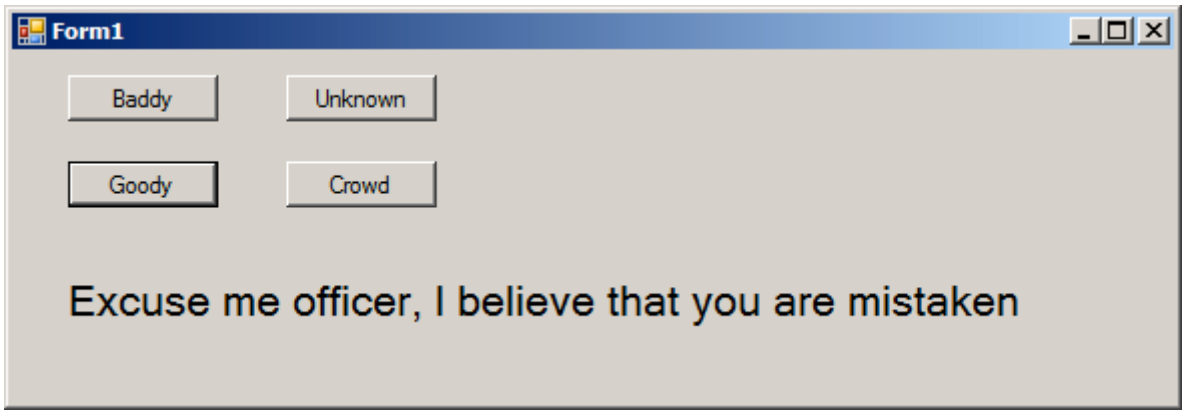
```
private Character target;
private Baddy mrEvil;
private Innocent passerBy;
```

Form_Load instantiates the sub classes:

```
private void Form1_Load(object sender, EventArgs e)
{
          mrEvil = new Baddy();
          passerBy = new Innocent();
}
```

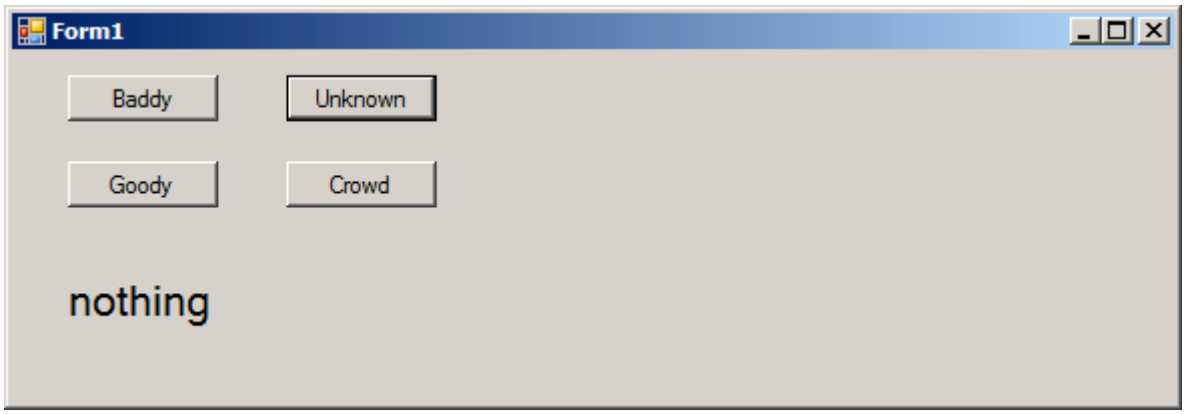
The Baddy and Goody Buttons call the hit() methods of mrEvil and passerBy.





The Unknown Button casts up to the Character class and calls hit()

```
target = (Character)mrEvil;
```



The Crowd Button creates an array of Characters, here 2 out of 10 are suspicious types.

```
Character[] theCrowd = new Character[10];
for (int x = 0; x < 10; x++)
{
                if (x % 4 == 0)
                    theCrowd[x] = new Baddy();
                else
                    theCrowd[x] = new Innocent();
}
```

The GetType() method will return what class an object belongs to.

```
if (theCrowd[x].GetType() == typeof(Baddy))
      mrEvil = (Baddy)theCrowd[x];
```

A loop can then run through the array of Characters, cast to the appropriate type and call hit(). Here a MessageBox is used to display hit() for each member in turn.