Polymorphism

Polymorphism, according to the dictionary, is *the condition of occurring in several forms*. In our programming context, it means that a parent class can have methods that may be overwritten by equally named methods in child classes, and even have empty methods in the parent class to force the structure into child classes.

A key benefit of polymorphism in programming is that it allows for the creation of uniform class structures in which child classes will be guaranteed to have the same methods structure, which, in turn, facilitates working with child classes, as they all will have the same method name, inputs and outputs, even if their behavior is different.

In our current *Eternal Quest* program, it makes it easier to add goals of different type into the same list and treat them all equally, even if the method behavior in them is different. It also makes it easier to expand the program functionality, as a different type of goal could later be added as a child class of the Goal superclass. Adding this would only require minimal coding changes and most of the added piece will be fully contained in the additional child class which is being forced into a specific structure.

The best example of polymorphism in the current program is in the `Record Event` method, which highlights the greatest behavioral difference between the child classes. In all cases, the code will return the number of points won by the recording of the event, but the actual procedure varies greatly.

Here is the parent class method:

```
    public virtual int RecordEvent()
    {
        _eventCount += 1;
        return _goalPoints;
    }
```

This is the same method from the SimpleGoal child class:

```
    public override int RecordEvent()

    {
```

```
        if(!_goalDone)
        {
            _goalDone = true;
            _eventCount += 1;
            return _goalPoints;
        } else {
            return 0;
        }
    }
```

Again, from the EternalGoal child class:

```
    public override int RecordEvent()
    {
        _eventCount += 1;
        return _goalPoints;
    }
```

and finally from the ChecklistGoal child class:

```
    public override int RecordEvent()
    {
        if(!_goalDone)
        {
            _eventCount += 1;
            if(_eventCount >= _targetEvents)
            {
                _goalDone = true;
                return _goalPoints + _bonusPoints;
            } else
            {
                return _goalPoints;
            }
        } else
        {
            return 0;
```

```
        }
    }
```