Polymorphism allows child objects to change or morph certain aspects of its inheritance. For example, my eternal goal program I have 3 different types of goals that inherit from a central goal class. Every goal object calculate how many points its worth each time it is complete, this is common across every type of goal. What is not common however is the HOW each goal object calculates the points. The simple goal type just counts its value, while an eternal goal type multiples its value by the number of times it has been completed. This is where polymorphism comes in, as the eternal goal type can override its inheritances of how to count its points. Yet we are still able to just call its parents class function of virtual GetPoints() and let the child decide how to calculate it.

The parent class sets a default of how to calculate points

    public virtual int GetPoints()

    {

        bool add = GetIsComplete();

            if ( add == true)

            {

                return \_value;

            }

        return 0;

    }

The eternal class can override the parent and calculate points in a different way

    public override int GetPoints()

    {

        return \_currentCount \* \_value;

    }

The checklist class also overrides how the points are calculated

    public override int GetPoints()

    {

        if (\_isComplete == true)

        {

            return ((\_currentCount \* \_value) + (\_bonusPoint));

        }

        return \_currentCount \* \_value;

    }