Polymorphism means the ability to change some things or settings in a class for different situations. When performing inheritance, attributes and methods are inherited from the base class to the child class. By using polymorphism, the methods may be modified to adapt to the needs of the child class.

Inheritance allows us to pass all attributes and methods to a child class, but not in all cases we need the child class to work exactly the same as the base class does. Thus, polymorphism is useful in such situations. We can modify the methods so that they can work in a way that suits the child class, which is different from the base class. Although the methods are different, we can still use inheritance to inherit other attributes or methods which work the same in both the base class and the child class. Moreover, with polymorphism, the override methods have the same names, which makes the code easier to write and read without making too many names and causing bugs and errors.

Consider a case where we need to create classes for calculating the volume of several solids, such as a sphere, a cone, a cube… They would need attributes such as the type of the solid, the colour of the solid… But for a sphere, it has a radius; for a cone, it has a radius, and a height; for a cube, it has a side length. All share some attributes, but require some extra. Moreover they all have different equations when calculating the volume. Thus, their GetVolume method will be different.

**public** **override** double GetArea()

    {

        double area = Math.PI \* Math.Pow(\_radius, 2);

        return area;

    }

**public** **override** double GetArea()

    {

        double area = Math.Pow(\_side, 2);

        return area;

    }

Above is an excerpt of a program which computes the area of some shapes. For example, the first GetArea method computes the area of a circle, while the second one computes the area of a square. Since both of them inherited the Shape class, which holds the original abstract method (not knowing the type of the shape, we could not compute its area), and the attributes such as the name of the shape. With polymorphism, we could initialize a suitable method for a shape, rather than having separate classes and initializing an independent one. This lowers the chance of errors and mistakes since the method must demonstrate similar functions to the original abstract method (same type of return, same name).