Hyacinth Chibueze Okoro: **Polymorphism**

In object-oriented programming, a line of code can sometimes behave in different ways depending on how it is being used or the context. This ability of a line of code to take different forms is known as polymorphism. The term polymorphism itself means “many forms” and it allows methods to be used in a way that depends on the object calling them.

One of the most important benefits of polymorphism is the ability of the coder to reuse or code reusability and flexibility. It allows developers to write more reusable code by enabling a single interface to work with different types of objects. This reduces code duplication and makes systems easier to extend and maintain.

Polymorphism is used by programmers in software design. For example, in our coding activity where we wrote a program that computes the areas of different shapes cut out of pieces of paper shown below;

public abstract class Shape

{

    private string \_color;

    public Shape(string color)

    {

        \_color = color;

    }

    public string GetColor()

    {

        return \_color;

    }

    public void SetColor(string color)

    {

        \_color = color;

    }

       public abstract double GetArea();

}

public class Circle : Shape

{

    private double \_radius;

    public Circle(string color, double radius) : base(color)

    {

        \_radius = radius;

    }

    // Notice the use of the override keyword here

    public override double GetArea()

    {

        return \_radius \* \_radius \* Math.PI;

In this example, the method GetArea() behaves in different ways for different classes that inherited the attributes of the Shape class. GetArea() will return a different for each of the classes like Circle, Rectangle and others because it will be overridden and made to behave the way the shape inheriting it wants it to behave.