* Explain the meaning of Polymorphism

Polymorphism is the ability to take on many forms. In programming, this principle is shown when one line of code can have different behavior depending on the context.

* Highlight a benefit of Polymorphism

Polymorphism is the fourth and crowning principle of programming with classes. Skillful use of abstraction, encapsulation and inheritance are all required to apply polymorphism effectively. The result is a simple but powerful mechanism for ensuring that programs are flexible and ready for change.

* Provide an application of Polymorphism

Calculate shape area

* Use a code example of Polymorphism from the program you wrote

Shape class

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();

}

Square class:  
public class Square : Shape

{

private double \_side;

public Square(string color, double side): base (color)

{

\_side = side;

}

public override double GetArea()

{

return \_side \* \_side;

}

}

Rectangle class:  
public class Rectangle : Shape

{

private double \_length;

private double \_width;

public Rectangle(string color, double length, double width): base (color)

{

\_length = length;

\_width = width;

}

public override double GetArea()

{

return \_length \* \_width;

}

}

Main Program:  
using System;

class Program

{

static void Main(string[] args)

{

List<Shape> shapes = new List<Shape>();

Square s1 = new Square("blue", 2);

shapes.Add(s1);

Rectangle s2 = new Rectangle("yellow", 3,5);

shapes.Add(s2);

Circle s3 = new Circle("Pink", 7);

shapes.Add(s3);

foreach(Shape s in shapes)

{

string color = s.GetColor();

double area = s.GetArea();

Console.WriteLine($"The {color} shape has an area of {area}.");

}

}

}