**Meaning of Polymorphism**

Polymorphism in object-oriented programming refers to the ability of objects to take on multiple forms. It allows a single interface or method to represent different underlying forms (data types). Specifically, polymorphism enables objects of different classes to be treated as objects of a common superclass, typically by overriding methods in child classes. This allows a more flexible and dynamic execution of code, where the same method call can behave differently based on the object type.

**Benefit of Polymorphism**

One major benefit of polymorphism is **code reusability and flexibility**. By using polymorphism, you can write more generic and flexible code that can work with different object types without modifying the code structure. For example, you can create a single method that handles multiple data types or class types, allowing the program to adapt to new changes with less modification.

**Application of Polymorphism**

Polymorphism is commonly applied in scenarios where multiple classes share a common method but have different implementations. For instance, in a program that involves various shapes (like circles, squares, and triangles), a common method such as CalculateArea() can be defined in a base class Shape, and each derived class can provide its own specific implementation of the method.

**Code Example of Polymorphism**

In your **Eternal Quest** project, where you're tracking different kinds of goals (simple, checklist, eternal goals), polymorphism can be used to handle the different types of goals in a unified way. Below is an example of how polymorphism can be applied in that context:

// Base class Goal

public abstract class Goal

{

public string Name { get; set; }

public int Points { get; set; }

// Abstract method - will be overridden in derived classes

public abstract void RecordEvent();

}

// Derived class for SimpleGoal

public class SimpleGoal : Goal

{

public override void RecordEvent()

{

Console.WriteLine($"Simple goal '{Name}' completed! You earned {Points} points.");

}

}

// Derived class for ChecklistGoal

public class ChecklistGoal : Goal

{

public int RequiredCount { get; set; }

public int CompletedCount { get; set; }

public override void RecordEvent()

{

CompletedCount++;

if (CompletedCount >= RequiredCount)

{

Console.WriteLine($"Checklist goal '{Name}' fully completed! You earned {Points} points.");

}

else

{

Console.WriteLine($"Progress on checklist goal '{Name}': {CompletedCount}/{RequiredCount} tasks completed.");

}

}

}

// Derived class for EternalGoal

public class EternalGoal : Goal

{

public override void RecordEvent()

{

Console.WriteLine($"Eternal goal '{Name}' recorded. Keep going! You earned {Points} points.");

}

}

// Polymorphism in action

public class Program

{

static void Main()

{

List<Goal> goals = new List<Goal>

{

new SimpleGoal { Name = "Run 5 miles", Points = 50 },

new ChecklistGoal { Name = "Read 5 books", Points = 100, RequiredCount = 5 },

new EternalGoal { Name = "Daily gratitude practice", Points = 10 }

};

// Loop through and record an event for each goal (polymorphism)

foreach (Goal goal in goals)

{

goal.RecordEvent(); // Calls the appropriate RecordEvent method based on the object's type

}

}

}