Polymorphism is a principle of coding where objects from different classes can be treated as objects of a common base class. This allows for code to be reusable, but also maintain flexibility as subclasses can override the base class to handle differences between the different classes. In my recent Eternal Quest project I was able to use polymorphism by having a base Goal class and then using SimpleGoal, EternalGoal and ChecklistGoal subclasses. Each of the subclasses had slight differences in the objects and methods, but had enough similarities that much of the code could be shared between them. In C# base classes can use virtual or abstract methods that can then be overridden by the subclasses to account for slight variations. Here is an example from my code:

Base Goal class:

public virtual void RecordEvent()

    {

        \_isComplete = true;

        Console.WriteLine($"Goal '{\_shortName}' completed! Description: {\_description}, Points: {\_points}");

        Console.WriteLine($"Total points earned: {\_points}");

    }

EternalGoal subclass:

public override void RecordEvent()

    {

        // override record event so that iscomplete is not set to true.

        Console.WriteLine($"Eternal goal '{\_shortName}' event recorded. Description: {\_description}, Points: {\_points}");

        Console.WriteLine($"Total points earned: {\_points}");

    }

ChecklistGoal subclass:

    public override void RecordEvent()

    {

        if (\_amountCompleted < \_target)

        {

            \_amountCompleted++;

            Console.WriteLine($"Checklist goal '{\_shortName}' updated. Completed: {\_amountCompleted}/{\_target}, Bonus: {\_bonus} points.");

            if (\_amountCompleted == \_target)

            {

                Console.WriteLine($"Checklist goal '{\_shortName}' completed! Total: {\_amountCompleted}/{\_target}. You earned an additional {\_bonus} points.");

                \_isComplete = true; // Mark as complete when target is reached

            }

        }

        else

        {

            \_amountCompleted++;

            Console.WriteLine($"Checklist goal '{\_shortName}' completed! Total: {\_amountCompleted}/{\_target}.");

}

By using polymorphism, I was able to call the methods for each of the subclasses much more simply allowing the program to be much easier to write and understand.