**Polymorphism** is an object-oriented programming principle that allows the same method name to work differently depending on the object calling it.  
The word “polymorphism” means “many forms.” In C#, we can have a **base class** with a method, and then **derived classes** override that method to give it their own special behavior.

A key benefit of polymorphism is **flexibility** our main code can work with the base class type and still run the correct version of a method for each specific goal type. This means we can add new goal types in the future without rewriting the main program logic.

In the **Eternal Quest** program, all goals inherit from the Goal base class. The base class has a RecordEvent() method, but each goal type (SimpleGoal, EternalGoal, ChecklistGoal) overrides it to behave differently. For example:

* SimpleGoal marks itself complete after one event.
* EternalGoal never completes and always gives points.
* ChecklistGoal tracks progress until a certain number of completions is reached.

**Code Example from Eternal Quest:**

Goal goal1 = new SimpleGoal("Read a book", "Finish one chapter", 50);

Goal goal2 = new ChecklistGoal("Exercise", "Go to the gym", 10, 5, 100);

goal1.RecordEvent(); // Runs SimpleGoal’s version of RecordEvent

goal2.RecordEvent(); // Runs ChecklistGoal’s version of RecordEvent

Even though both are stored as Goal, each one’s own RecordEvent is called.

**In short**, polymorphism makes our Eternal Quest program easier to expand, because we can add new goal types without changing how the rest of the program calls the methods.