**Polymorphism**

Polymorphism is a fundamental concept in object-oriented programming that allows objects of different classes to be treated as objects of a common superclass. It enables a single interface to be used for entities of various types, promoting flexibility and extensibility in software design.

One key benefit of polymorphism is enhanced code reusability and maintainability. By programming to interfaces rather than concrete implementations, polymorphism facilitates the creation of modular and scalable codebases.

A practical application of polymorphism is in method overriding, where subclasses provide their own implementation of a method declared in a superclass. For example, in our goal-tracking program, each type of goal (SimpleGoal, EternalGoal, etc.) overrides the RecordEvent method to provide behavior specific to that goal type.

Here's a simplified code example from our program:

class Goal

{

public virtual void RecordEvent()

{

Console.WriteLine("Event recorded for goal.");

}

}

class SimpleGoal : Goal

{

public override void RecordEvent()

{

Console.WriteLine("Event recorded for simple goal.");

}

}

class EternalGoal : Goal

{

// No need to override RecordEvent for eternal goals

}

In this example, the RecordEvent method is polymorphic, as it can behave differently based on the specific type of goal object calling it. This flexibility allows for the seamless integration of various goal types within our program.