**Natalie Kolbo**

**CSE210**

**W06 Assignment: Explain Polymorphism**

**What is Polymorphism and Why is it Important?**

**Polymorphism is a fundamental principle of Object-Oriented Programming (OOP) that allows objects of different classes to be treated as objects of a common superclass. This means that a single interface (such as a method in a base class) can be used to call different implementations depending on the specific subclass.**

**One of the biggest benefits of polymorphism is that it promotes flexibility and scalability in software design. By allowing different classes to override shared methods, polymorphism makes it easier to extend functionality without modifying existing code. This follows the Open-Closed Principle (OCP), meaning a system is open for extension but closed for modification—reducing maintenance time and the risk of breaking other parts of the code.**

**In real-world applications, polymorphism is widely used in frameworks, UI design, and game development. For example, in a payment processing system, a single method ProcessPayment() could handle different payment types (CreditCard, PayPal, Cryptocurrency) without needing separate function names.**

**Application of Polymorphism in My Eternal Quest Program**

**In my Eternal Quest Program, I applied polymorphism by creating a base class (Goal) with a method RecordEvent(), and then overriding this method in multiple subclasses (SimpleGoal, EternalGoal, and ChecklistGoal). Each subclass provides a different implementation of RecordEvent() to handle unique goal-tracking behaviors.**

**Here’s an example of polymorphism in action:**

**🔹 Goal.cs (Base Class)**

**public abstract class Goal**

**{**

**protected string \_name;**

**protected int \_points;**

**public Goal(string name, int points)**

**{**

**\_name = name;**

**\_points = points;**

**}**

**public abstract void RecordEvent(); // Abstract method to enforce polymorphism**

**}**

**🔹 SimpleGoal.cs (Derived Class)**

**public class SimpleGoal : Goal**

**{**

**private bool \_isComplete;**

**public SimpleGoal(string name, int points) : base(name, points)**

**{**

**\_isComplete = false;**

**}**

**public override void RecordEvent()**

**{**

**\_isComplete = true;**

**Console.WriteLine($"✅ Goal '{\_name}' completed! You earned {\_points} points.");**

**}**

**}**

**🔹 ChecklistGoal.cs (Another Derived Class)**

**public class ChecklistGoal : Goal**

**{**

**private int \_timesCompleted;**

**private int \_requiredCompletions;**

**private int \_bonusPoints;**

**public ChecklistGoal(string name, int points, int requiredCompletions, int bonusPoints)**

**: base(name, points)**

**{**

**\_timesCompleted = 0;**

**\_requiredCompletions = requiredCompletions;**

**\_bonusPoints = bonusPoints;**

**}**

**public override void RecordEvent()**

**{**

**\_timesCompleted++;**

**Console.WriteLine($"📋 Progress on '{\_name}': {\_timesCompleted}/{\_requiredCompletions} completed.");**

**if (\_timesCompleted >= \_requiredCompletions)**

**{**

**Console.WriteLine($" Goal '{\_name}' fully completed! Bonus {\_bonusPoints} points awarded!");**

**}**

**}**

**}**

**Polymorphism in Action (Calling the Method on Different Objects)**

**Because of polymorphism, we can store different goal types in a single list and call RecordEvent() without needing to know which specific subclass we are dealing with:**

**List<Goal> goals = new List<Goal>();**

**goals.Add(new SimpleGoal("Run a marathon", 1000));**

**goals.Add(new ChecklistGoal("Attend temple", 50, 10, 500));**

**foreach (Goal goal in goals)**

**{**

**goal.RecordEvent(); // Calls the overridden method from the correct subclass!**

**}**

**Conclusion**

**Polymorphism is critical in OOP because it allows flexibility, scalability, and cleaner code by enabling different objects to share a common interface while implementing their own behavior. In my Eternal Quest Program, I leveraged polymorphism through method overriding, allowing each goal type to customize how events are recorded without modifying the rest of the system.**