## Polymorphism

Polymorphism is a concept in object-oriented programming that lets you use a single interface to represent different types of objects. Imagine having a general "goal" that can be a simple goal, an eternal goal, a checklist goal, or even a negative goal. Polymorphism lets you treat all these different goals in the same way, even though they might behave differently.

### Meaning of Polymorphism:

Polymorphism means "many forms." In programming, it allows different classes to be treated as if they are objects of a common superclass. This way, you can call the same method on different objects and each object can respond in its own way.

### Benefit of Polymorphism:

A big benefit of polymorphism is flexibility. It makes your code easier to extend and maintain. For example, you can add new types of goals without changing the code that uses them. This means less work and fewer errors when you need to update your program.

### Application of Polymorphism:

Polymorphism is used in many applications where you need to work with objects that share a common interface. For example, in our goal-tracking program, we can create different types of goals (simple, eternal, checklist, negative) and handle them all through a common interface.

### Code Example from the Program:

1. Base Class: Goal

*public abstract class Goal  
{  
 public string ShortName { get; set; }  
 public string Description { get; set; }  
 public int Points { get; set; }  
  
 protected Goal(string name, string description, int points)  
 {  
 ShortName = name;  
 Description = description;  
 Points = points;  
 }  
  
 public abstract void RecordEvent();  
 public abstract bool IsComplete();  
 public abstract string GetDetailsString();  
 public abstract string GetStringRepresentation();  
}*

2. Derived Classes: SimpleGoal, EternalGoal, ChecklistGoal, NegativeGoal

*public class SimpleGoal : Goal  
{  
 public bool IsCompleted { get; set; }  
  
 public SimpleGoal(string name, string description, int points)  
 : base(name, description, points)  
 {  
 IsCompleted = false;  
 }  
  
 public override void RecordEvent()  
 {  
 IsCompleted = true;  
 }  
  
 public override bool IsComplete()  
 {  
 return IsCompleted;  
 }  
  
 public override string GetDetailsString()  
 {  
 return $"[{(IsCompleted ? "X" : " ")}] {ShortName}: {Description} - Points: {Points}";  
 }  
  
 public override string GetStringRepresentation()  
 {  
 return $"SimpleGoal:{ShortName},{Description},{Points},{IsCompleted}";  
 }  
}*

3. Using Polymorphism in GoalManager:

*public void RecordEvent()  
{  
 while (true)  
 {  
 Console.Clear();  
 DisplayAppName();  
  
 if (\_goals.Count == 0)  
 {  
 Console.WriteLine("No goals available to record an event.");  
 PressAnyKeyToContinue();  
 return;  
 }  
  
 try  
 {  
 Console.WriteLine("Select the goal to record an event for:");  
 List<int> incompleteGoalIndices = new List<int>();  
  
 for (int i = 0; i < \_goals.Count; i++)  
 {  
 if (!\_goals[i].IsComplete())  
 {  
 Console.WriteLine($" {incompleteGoalIndices.Count + 1}. {\_goals[i].ShortName}");  
 incompleteGoalIndices.Add(i);  
 }  
 }  
  
 if (incompleteGoalIndices.Count == 0)  
 {  
 Console.WriteLine("No incomplete goals available.");  
 PressAnyKeyToContinue();  
 return;  
 }  
  
 Console.Write("  
Enter the goal index to record event: ");  
 int goalIndex;  
 if (!int.TryParse(Console.ReadLine(), out goalIndex) || goalIndex < 1 || goalIndex > incompleteGoalIndices.Count)  
 {  
 Console.WriteLine("  
Invalid goal index. Please enter a valid number.");  
 PressAnyKeyToContinue();  
 continue;  
 }  
  
 Goal goal = \_goals[incompleteGoalIndices[goalIndex - 1]];  
 goal.RecordEvent();  
 int pointsEarned = goal.Points;  
  
 if (goal is ChecklistGoal checklistGoal && checklistGoal.IsComplete())  
 {  
 pointsEarned += checklistGoal.Bonus;  
 \_score += pointsEarned;  
 Console.WriteLine($"  
Goal completed! You earned {pointsEarned} points.");  
 DisplayCelebrationEffect();  
 }  
 else if (goal is NegativeGoal)  
 {  
 \_score += pointsEarned;  
 Console.WriteLine($"  
Bad habit recorded! You lost {-pointsEarned} points.");  
 }  
 else  
 {  
 \_score += pointsEarned;  
 Console.WriteLine($"  
Congratulations! You earned {pointsEarned} points for recording this event.");  
 DisplayCelebrationEffect();  
 }  
  
 PressAnyKeyToContinue();  
 break;  
 }  
 catch (FormatException)  
 {  
 Console.WriteLine("  
Invalid input. Please enter a valid number.");  
 PressAnyKeyToContinue();  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine($"  
An error occurred: {ex.Message}");  
 PressAnyKeyToContinue();  
 }  
 }  
}*

In this example, Goal is the base class, and SimpleGoal, EternalGoal, ChecklistGoal, and NegativeGoal are subclasses. Each subclass implements its own version of the RecordEvent and GetDetailsString methods. The RecordEvent method in GoalManager uses polymorphism to handle all these different types of goals through a single interface, making the code flexible and easy to maintain.

By using polymorphism, I can add new types of goals in the future without changing the code that interacts with goals. This makes my program more flexible and easier to extend.