**Explanation of Inheritance**

## What is Inheritance?

Inheritance is a key concept in object-oriented programming (OOP). It allows one class (the child class) to use properties and methods from another class (the parent class). This means the child class gets the features of the parent class automatically.

## Benefit of Inheritance

One main benefit of inheritance is reusing code. Instead of writing the same code over and over, you can write it once in a parent class and use it in multiple child classes. This makes your code cleaner and easier to maintain.

## Application of Inheritance

Inheritance is useful when you have a hierarchy. For example, in a game, you might have a general Character class. Then, you can create specific types of characters like Warrior, Mage, and Archer that inherit from Character. These specific characters get the basic features from Character but can also have their own unique features.

## Code Example from the Program

In my mindfulness app program, I used inheritance to define different activities. Here’s an example:

***// My class Activity*** *public abstract class Activity  
{  
 private string \_name;  
 private string \_description;  
 protected int \_duration;  
  
 protected Activity(string name, string description)  
 {  
 \_name = name;  
 \_description = description;  
 }  
  
 public void DisplayStartingMessage()  
 {  
 Console.Clear();  
 Console.WriteLine($"Welcome to the {\_name} Activity\n");  
 Console.WriteLine(\_description);  
 Console.Write("\nPlease specify the session duration in seconds: ");  
 while (!int.TryParse(Console.ReadLine(), out \_duration) || \_duration <= 0)  
 {  
 Console.Write("Please enter a valid positive number for the duration: ");  
 }  
 Console.Clear();  
 Console.WriteLine("Prepare to begin...");  
 ShowSpinner(5);  
 Console.WriteLine("\n");  
 }  
  
 public void DisplayEndingMessage()  
 {  
 Console.WriteLine("\n\nActivity Complete!!");  
 ShowSpinner(5);  
 Console.WriteLine($"\nYou have completed the {\_name} Activity for {\_duration} seconds.");  
 ShowSpinner(4);  
 }  
  
 protected void ShowSpinner(int seconds)  
 {  
 string[] spinner = { "/", "-", "\\", "|" };  
 for (int i = 0; i < seconds \* 4; i++)  
 {  
 Console.Write(spinner[i % 4]);  
 Thread.Sleep(250);  
 Console.Write("\b");  
 }  
 Console.Write(" ");  
 Console.Write("\b");  
 }  
  
 protected void ShowCountDown(int seconds)  
 {  
 for (int i = seconds; i > 0; i--)  
 {  
 Console.Write($"{i} ");  
 Thread.Sleep(1000);  
 Console.SetCursorPosition(Console.CursorLeft - (i.ToString().Length + 1), Console.CursorTop);  
 Console.Write(new string(' ', i.ToString().Length + 1));  
 Console.SetCursorPosition(Console.CursorLeft - (i.ToString().Length + 1), Console.CursorTop);  
 }  
 Console.WriteLine();  
 }  
  
 public abstract void Run();  
}*  
**// My class BreathingActivity** *public class BreathingActivity : Activity  
{  
 public BreathingActivity() : base("Breathing", "This activity will help you relax by walking you through breathing in and out slowly. Clear your mind and focus on your breathing.")  
 {  
 }  
  
 public override void Run()  
 {  
 DisplayStartingMessage();  
  
 Stopwatch stopwatch = new Stopwatch();  
 stopwatch.Start();  
  
 while (stopwatch.Elapsed.TotalSeconds < \_duration)  
 {  
 Console.Write("\nBreathe in... ");  
 ShowCountDown(4);  
 if (stopwatch.Elapsed.TotalSeconds >= \_duration) break;  
  
 Console.Write("Now breathe out... ");  
 ShowCountDown(6);  
 }  
  
 stopwatch.Stop();  
 DisplayEndingMessage();  
 }  
}*

In this example:  
- The `Activity` base class has common properties and methods for all activities, like `\_name`, `\_description`, `\_duration`, `DisplayStartingMessage()`, `DisplayEndingMessage()`, `ShowSpinner()`, and `ShowCountDown()`.  
- The `BreathingActivity` class inherits from `Activity` and has its own `Run()` method, which is specific to the breathing exercise.  
  
Using inheritance here makes it easy for me to create new activities by inheriting from `Activity` and adding or changing what I need for each specific type. This shows how inheritance helps in reusing code.