# Inheritance Explanation

Inheritance is a core idea in object-oriented programming that lets one class build on another.   
When a child class inherits from a parent class, it automatically gets access to the parent’s methods and properties.   
This setup is perfect for modeling “is-a” relationships. For example, a BreathingActivity is an Activity,   
so it makes sense for it to inherit from the Activity class.  
  
In my program, I created a base class called Activity that handles shared functionality like displaying start and end messages,   
managing durations, running countdowns, and showing spinner animations. Then I built specific activity types: BreathingActivity,   
ReflectingActivity, and ListingActivity as subclasses. These subclasses inherit everything from Activity, but each one adds its own twist.   
For instance, BreathingActivity walks the user through timed breathing exercises, while ReflectingActivity prompts thoughtful questions,   
and ListingActivity challenges users to list responses to a random topic.  
  
One of the biggest advantages of inheritance is that it makes code cleaner and easier to maintain. I only had to write the shared logic once in the Activity class.   
If I ever improve the countdown or spinner, all the activities benefit instantly with no need to update each one separately. It also makes the program flexible:   
adding a new activity is as simple as creating a new subclass.  
  
Here’s a snippet from my code that shows inheritance in action:

if (choice == "1") new BreathingActivity().Run();  
else if (choice == "2") new ReflectingActivity().Run();  
else if (choice == "3") new ListingActivity().Run();