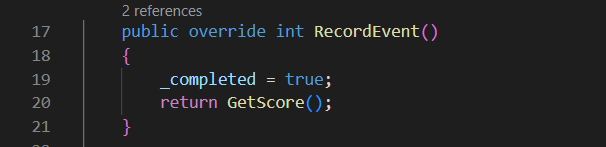
Polymorphism is the ability to use and alter behaviors from the parent class within a child class to meet the needs of the program. One benefit of Polymorphism is the reusability and the flexibility of it. It prevents repetition as you don’t have to write the same code over and over again. It also gives you the ability to override/change the method of the parent class if the functionality of the inherited method in the child class differs in any way, without having to create a completely new method. These methods can be used and altered in as many child classes is needed. An application of Polymorphism could be shown with a program about shapes. The parent class would be “Shapes” and would hold methods like “CalculateArea,” “CalculatePerimeter,” and so on. It could include the keyword “virtual” and have default code placed in the body of it, or it could use the keyword “abstract,” which would require that the method doesn’t have a body at all. Some child classes may include “Circle,” “Square,” “Rectangle,” etc. All of these could then inherit the “Shapes” class and use its methods. For example, “Square,” could override “CalculatePerimeter” and put the specific calculation for a square within the body of the abstract method inherited from the “Shapes” class.

A screen shot of a computer program

Description automatically generatedIn this example from my code, I have the parent class of “Goal.” It holds the abstract method “RecordEvent.” This method was used in all of my goal subclasses. The one I’ve shown below is “SimpleGoals.” I inherited and overrode “RecordEvent” from the parent class to meet the functionality requirement of that particular method in the “SimpleGoals” class. I could then call “RecordEvent” in my main program class and it would grab the code from whichever class it needed it from.