Version Control

Version control allows multiple people to collaborate on a project in an organized way. There are tools such as GitHub that enable teams to start projects, monitor changes and manage contributions effectively. It tracks and keeps a history of changes made to files, making updates without overwriting the same code.

One benefit is that it ensures a clear record of all modifications, making it easy to revert to previous versions if needed. An example of a command used in version control is *git commit -m* "made changes to the code". This command saves changes to the code along with a message describing what was updated.

Abstraction

We use abstraction in programming to simplify complex ideas. One of its benefits is to keep the code organized and easy to work with. An application of abstraction is using classes as a template of the code, so we are abstracting away the ideas of how the code is going to be structured. I used 4 classes in my journal program to organize my ideas. An example is how I call the class methods inside another class, such as the journal methods (journal.DisplayAll, journal.LoadFromFile, etc.) inside the program class. Thus, keeping the code organized and readable.

Encapsulation

In object-oriented programming, encapsulation is the practice of hiding or restricting direct access to attributes, member variables, or methods within a class, and allowing interaction only through public interfaces. This is done to protect the internal state of an object and prevent unwanted changes that could break the code.   
A common problem in programming is unauthorized changes to data, which can affect the program's output. For example, if I change the "\_isHidden" variable inside my Word class to "public", then other parts of the program could change it freely, for instance from false to true before the scripture is displayed. This could ruin the output by making the entire scripture appear blank from the beginning. To prevent this, I encapsulated my member variable so it can't be accessed or changed from outside the class.  
One of the benefits of encapsulation is that it helps organize the code clearly into manageable sections, where each class is responsible for its own behavior. It also allows future changes to be made safely without affecting the program's output.

Inheritance

Inheritance allows us to create a new class that uses the behavior defined in another class, thus inheriting the properties and methods of that class. The original class is called the parent or base class, and the new class is called the child or derived class. One benefit of inheritance is code reusability. Instead of rewriting the same logic, we use the code once and later on reuse it from the parent class. Thus, keeping the code cleaner, easier to update and more organized.  
In C#, inheritance is done using the : symbol. For example, in my program, I created a base class called *Activity,* which contains shared behavior like StartActivity(), EndActivity() and CountDown(). Then, I created other classes such as BreathingActivity, ReflectionActivity and ListingActivity that inherit behavior from the *Activity* Class.   
One example is *public class ListingActivity : Activity*. This means that *ListingActivity* gets access to all the public and protected methods and variables defined in Activity. Inheritance helps reduce code duplication and allows us to build a hierarchy of classes with shared functionality.