# Abstraction

Abstraction in C# is a way to simplify complex systems by hiding unnecessary implementation details and exposing only the essential features to the user. I used abstraction in a quiz application where the `GetQuestion` method in the `RandomQuiz` class retrieves a random question from a list. This method hides the details of how the list is managed and populated, focusing instead on the essential feature of retrieving a question. This abstraction makes the code easier to maintain and extend, as future changes to the way questions are stored or retrieved won't require modifications to the rest of the program.

# Encapsulation

The primary purpose of encapsulation is to protect data from unauthorized access and modification. This approach safeguards the integrity of the data, preventing external code from arbitrarily altering the data. Encapsulation makes the program more secure and reliable, as it controls how data is accessed and modified. For example, in my code, I encapsulated the variables `book`, `chapter`, `verse`, and `endVerse` inside the class `Reference` by making them private and providing public methods to interact with these variables. This ensures that these variables can only be accessed and modified through the methods I defined, maintaining the integrity of the data.

Encapsulation was applied in my program to protect the data within the Reference class. I made the variables book, chapter, verse, and endVerse private, ensuring they could only be accessed or modified through public methods like constructors and getters. This approach safeguards the integrity of the data and prevents external code from making unauthorized changes. Encapsulation makes the program more flexible because any future changes to how the data is stored or validated can be implemented within the Reference class without impacting other parts of the application.

# Inheritance

# Inheritance is a fundamental concept in programming that allows a class to inherit properties and behaviors from another class. This means that the subclass can reuse code from the superclass, which promotes code reusability and organization. By inheriting common properties and methods from a base class, multiple subclasses can share the same code, making the program more maintainable and less prone to errors. Inheritance can be applied in a scenario where you have a base class containing general information and behavior, such as a `Student` class with properties like `Name` and methods like `SubmitAssignment`. Subclasses could then inherit from `Student`, such as `MathStudent` or `WritingStudent`, each with additional specific properties and methods.

Inheritance was used to promote code reusability in my program. I created a base Assignment class with common properties like studentName and topic, and then derived a WritingAssignment class from it. The WritingAssignment class inherited these properties and added its own specific features, such as a title. This use of inheritance reduces redundancy and makes the program more maintainable. For example, if I need to add a new property like dueDate to all assignments, I can add it to the base Assignment class, and all derived classes will automatically inherit it.

# Polymorphism

Polymorphism is the ability of a single line of code to exhibit different behaviors based on the context in which it is used. This flexibility allows for more dynamic and versatile code, making it easier to maintain and extend. One significant benefit of polymorphism is that it allows for changes in code behavior with minimal modifications. This means that a single line of code can be adapted to perform different functions, enhancing code reusability and reducing the need for extensive rewrites.

Polymorphism was utilized in my program to calculate the area of different shapes. I created a base Shape class with a virtual GetArea method, and then overrode this method in derived classes like Circle and Rectangle. This allows a single method call (GetArea) to exhibit different behaviors depending on the shape being used. Polymorphism makes the program more flexible because new shapes can be added in the future by simply creating new classes that override the GetArea method, without modifying the existing code.