Principles of Programming with classes:

There are four basic principles to programming with class.

1. **Abstraction:** *Abstraction can simply be defined as the way we turn complex ideas into simple codes. It is the first principle in programming with classes. It makes it easy to talk about codes with both programmer and non-programmers. It allows developers to focus more on what an object does rather than how it does it. It reduces the complexity of codes and increase the usability.*

*I used it while working on the scripture memorizer program by creating a classed named scripture that managed the text and logic for hiding words and the main program only had to interact with the class through public methods.*

*This way, using it made it easier to modify the way words are hidden without touching or affecting the main program.*

1. **Encapsulation:** *Encapsulation can basically be defined as the way we carefully hid or enclose the details of the behavioral attributes of our classes in such a way that they can’t be affected, seen or adjusted by other codes. Encapsulation is a very important part of our programming as it helps us to reduce the rate at which different parts of our codes depends on each other and it protects them from breaking changes.*

*I used Encapsulation while working on the online ordering program, I had to create a class named order that had a private member variables and to access these variables, and you will need to do that using public method*

*It improves flexibility in a way that other external codes could modify data in an unintended way*

1. **Inheritance:** *Inheritance allows a class to derive properties and behavior from another class, promoting code reuse and hierarchy organization*

*I used inheritance while working on my eternal quest program and I created a base file called Goal that contains common attributes like name and descriptions. Then I used it to derive different goal types from the base class with each one of them having their unique implementations.*

*It improves flexibility by making it easier to add up more classes without tampering with the existing code.*

1. **Polymorphism:** Polymorphism can be defined as the concept that allows object of different classes to take on many forms and to be able to see this, there is need to understand method overriding. This is the ability of a child class to override or adjust the behavior from a parent class. It is the fourth and crowning principle of programming with class. It requires abstraction, encapsulation and inheritance to function properly. The advantage of polymorphism is that it promotes the reusability and maintenance of codes and also makes it easier to extend and modify our programs without changing the currently existing program.

I used Polymorphism while working on a mindfulness program by creating a base activity class with a virtual run activity method. Each specific activity overrode this method to provide different experiences while following a common structure

It improves flexibility by allowing the usage of different activities without changing the program’s main logic

By applying these four principles—abstraction, encapsulation, inheritance, and polymorphism—I made my programs more modular, maintainable, and scalable. These principles not only helped me organize my code better but also prepared me for writing flexible and efficient software in future projects.