**Final Project Articulate Activity**

**Abstraction**

Abstraction is a fundamental concept that refers to the ability to represent and handle complex concepts in a simplified way. It is a technique that allows you to hide unnecessary or complex details of an entity to focus only on aspects relevant to the context in which you are working, it allows programmers to interact with components and objects of a program more easily, since they do not need to know all the internal details of how those components work.

This principle was used in my project, because the idea of keeping track of the comments or list of comments for each video was complex. Therefore through Abstraction I thought of simplified solutions that can help me make the program meet its goal. What I did was focus on what each video should have and how it should be displayed, so I made a class called Video, Comments, etc. This helped me separate the necessary details from the unnecessary ones.

**Encapsulation**

Encapsulation is a principle of object-oriented programming that refers to the intentional hiding of an object's internal details and controlled exposure of its public interface. That is, encapsulation allows the data and related methods of an object to be grouped into a cohesive unit and protected from unauthorized access from outside that object.

**Inheritance**

Inheritance allows one class (called a derived class or subclass) to acquire the properties and behaviors of another class (called a base class or superclass). In other words, inheritance allows you to create a new class based on an existing class, thus reusing the code and features of the base class, when one class inherits from another, the derived class inherits all the fields and methods (properties and functions) of the base class and can add new fields and methods, or even modify and extend existing ones. This means that the derived class has access to the public and protected features of the base class and can use them as if they were its own.

**Polymorphism**

Polymorphism is a concept that allows the same function name or identifier to have multiple different behaviors or meanings, depending on the context in which it is used. In other words, polymorphism allows different classes or types of objects to respond to the same function call in specific and appropriate ways based on their own behavior.