* **Explain the meaning of each principle.**

Abstraction – It is the method we use to simplify tasks or in other words when We have a project, this project might have different tasks, so we can divide these tasks into classes so that the program might be easy to code, debug etc. Then we can combine all this tasks into one program.

Encapsulation – Is when we hide certain elements of each class from other classes, this can allow us to use appropriate ways to get information from other classes (getters and setters). During my experience on encapsulation, I notice that by preventing me to access other classes’ elements, it helps me to be more organized in my code.

Inheritance – Is when a class inherits attributes form another class, by setting a base class and derived classes, where the derived classes inherit attributes from the base class, usually those attributes are similar behaviors that can be stored in the base class then have the derived classes inherit it.

Polymorphism – Is when a line of code can offer different behaviors. When certain classes have similar behaviors, and maybe slight differences, the derived classes can inherit those behaviors from the base class and override it for its own.

* **Highlight how each principle was used in your final project.**

During the project I made, we can observe Abstraction when I divided my portfolio project in different tasks or classes: **Portfolio Class, ManagePortfolio Class, ClassesUncompleted Class, CurrentClasses Class, ClassesCompleted Class, OwnedBook Class, StudyTopics Class** and the **WantedBooks Class.** Each of those classes perform their on tasks and they can run by themselves without any problem allowing it to be easy to debug or change anything I want in the future.

While in the encapsulation part, I was able to use the private code on every member variable to protect all the member variables from each class, I also used getters and setters to access some of them.

Inheritance in my project was very essential to use because most of the classes had similar behaviors that I needed to inherit from the base class, the constructors that where very essential to define elements for each class.

One other essential thing was the Polymorphism that allowed me to override certain member functions from the base class to the derived classes and each class using the same displaying functions but with different outcomes.

* **Explain how these principles make your final project more flexible for changes.**

Using these concepts, I can add new tasks, and behaviors later int my program without having to change the whole code later in life and the program can we repaired, changed and updated all in parts.

* **Thoroughly explain these concepts (this likely cannot be done in less than 100 words)**

In this case the portfolio program I created has a section that shows the personal info of the user, the school requirements vs the users achievements based on his grades of the classes he entered, to point out that the program calculates the gpa, letter grade on its own, it show the classes completed, current and the future ones in a different way, and the user also can input a list of books he want to get, books he already has and a list of topics to study in the future and all this he can save as a file and load it later.

These complexities are so easy to handle with programming with classes where all is coded in sections and each part can still run without the other.