Object-Oriented Principles

1. Divide and Conquer Principle
* Divide overall program into several objects that will interact with each other to solve a problem.
* Complicated and complex transaction is divided into smaller, and more manageable tasks.
1. Encapsulation Principle
* Decide what attribute it has and what actions it will take.
* The goal is to make sure each object is a self-contained module with a clear responsibility. The attributes and actions are both necessary to carry out their own roles.
* Each object should know how to solve its task and has the information needed.
* Bundling of data with methods that operate it into a single unit.
* It is being used in the form of classes.
* Restrict direct access to some components of an object.
1. Interface Principle
* Objects need to be clarified exactly on how they can interact with one another for them to work cooperatively and efficiently
* Public interface should be present clearly to determines how other objects being used.
* No code should be forced to depend on methods that does not use.
1. Information Hiding Principle
* Certain details of object’s individual design and performance should be hidden from other objects to enable them to work together cooperatively.
* Each object should protect their users from unnecessary details of how they perform their own role.
1. Generality Principle
* To make object as generally as useful as possible, they need to be design not for a particular task, instead for a particular kind of task.
* Software libraries underlies this principle.
* Objects designed to solve a kind of task rather than a singular task.
1. Extensibility Principle
* The ability to extend an object’s behaviour to handle new tasks without too much difficulty or hardship strengthen the use of OOP.
* Object should be designed so that their functionality can be extended to carry out more specialized task.
* Provide a new functionality to the code in a straightforward manner.
* For example, reusable classes and techniques