Lab Report (OO Continued)

Name: Vincent Cendana
Red-Id: 133417528

Reflection:

Explain how design pattern help write simpler code, what do your understand by code maintenance and clean code

Design patterns help to write simpler code because they are guidelines toward creating expandable and reusable code. Design patterns are techniques that can be used to structure common ideas found in programming. For example, the Observer pattern is used commonly when programmers want to have objects be listening in on another object's changes. While this pattern isn't used in every program, it is common enough that programmers have identified its use and characteristics. These techniques are optimal for creating scalable and reusable programs. For example, with the Template pattern, you can hypothetically subclass it as much as you desire because of how flexible it is. You don't have to edit the internal code of the Template class, instead you can just extend it. This follows the **open for extension, closed for modification** principle.

Code maintenance means whenever the programmer has to edit or improve code after it has been written. Typically, maintenance is required whenever there is a bug or new features have to be updated/added. Writing clean code means to write code that is easy to understand and modify. An example of unclean code would be having to modify the implementation of a base class rather than being able to extend it.

Design patterns may also allow for clean code and easier code maintenance. Say for example you want to only always have one instance of a class; You may accidentally create a new instance and be left wondering what went wrong in your code. There may be no errors thrown, but the output could be wrong. Design patterns can help prevent errors like this from happening. In this case, you can use a Singleton pattern to stop the programmer from accidentally making new instances of the class. This is because the constructor is marked as private and attempted outside usage will throw an error.

Answer the following patterns

1) Observer

The Observer pattern is used for when you want one object to be made aware of another object's changes. For example, you would want a Graph object to be made aware of changes to a Data object. The Observer class usually has a method (.notify(…)) that is called by a Subject class. This method is usually passed arguments that represent the change in state. The Subject class is the class that the Observer watches. The Subject class is aggregated of Observer objects and calls their .notify methods whenever their state changes.

2) Template

The Template pattern is useful for when multiple different classes follow the same steps with the same result, but they implement them differently. The Template pattern defines a base class that will define abstract methods that will be called in the template method. The template method is a defined method in the base class

that calls all the abstract methods together. Subclasses of the base class will implement the abstract methods differently but will return the same type. When the template method on the subclass is called, the subclass' overridden methods will be called within the method.

3) Singleton

The Singleton pattern is useful for you only want instance of a class to exist at one time. The Singleton pattern works by marking the constructor as private, so that the program cannot use the new keyword to make a new one.. Instead, it exposes a getInstance method which will return the instance of the Singleton. A static instance variable **instance** is made the same type as the Singleton. The getInstance method works by checking if **instance** is null. If it is, then it makes **instance** equal to a new Singleton instance. After, it will return this instance. That way, the same instance is returned by the method every time.