**DESIGN PATTERNS**

Adapter :-

Adapter pattern works as a bridge between two incompatible interfaces.

*Usage:-*  It helps to make existing classes work with others without modifying their

source code.

Decorator:-

It allows a user to add new functionality to an existing object without altering its structure.

*Usage:-*  Ex: From a base text, several formats can be applied cumulatively and in any order (Bold, Italic, Underline, Strike-through, Color…).

Facade:-

It hides the complexities of the system and provides an interface to the client using which the client can access the system.

*Usage:-* A segment of the client community needs a simplified interface to the overall functionality of a complex subsystem.

Factory:-

It says that just define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate.

*Usage:-*  It is used for creating objects to encapsulate the instantiation logic. Client doesn’t know the actual instantiation logic of entity

Iterator:-

the iterator pattern is a design pattern in which an iterator is used to traverse a container and access the container's elements.

*Usage:-* It is used to traverse a collection object elements one by one

Memento:-

Memento is a behavioral design pattern that lets you save and restore the previous state of an object without revealing the details of its implementation.

*Usage:-* Memento Design Pattern will be used in situations where some actions are undoable, therefore requiring to rollback to a previous state

Observer:-

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its depenedent objects are to be notified automatically.

*Usage:-* It is used when an action needs to be performed when an event/change occurs.

Prototype:-

Prototype pattern says that cloning of an existing object instead of creating new one and can also be customized as per the requirement

*Usage:-* It is used when the Object creation is a costly affair and requires a lot of time and resources and you have a similar object already existing.

Proxy:-

It creates proxy for original object and provides the control for accessing the original object.

*Usage:-* It acts as an authorization layer to verify that whether the actual user has access the appropriate content or not.

Singleton:-

The Singleton pattern ensures that a class has only one instance and provides a global point of access to that instance.

*Usage:-* EX: In Java the Singleton pattern will ensure that there is only one instance of a class is created in the Java Virtual Machine. It is used to provide global point of access to the object.