**Design Patterns**

* This design pattern is used at the time of creating or scaling your application irrespective of which programming language you are using.
* Every design pattern is classified into some type

1)Creational Design Pattern(builder,Singleton,Factory)

2)Structural Design Pattern ()

3)Behavioural Design Pattern (): How to Object interact with each other it will decide

**1)Single design Pattern**

* in this design pattern basically, we create a single object and we are reusing this object based on our requirements.
* here we are making the constructor private and making a static method that returns the same class object either early or lazy loading.
* so if we are using our application in the multithreaded environment using ExecutorService executer=Executor().newFixedThreadPool(2)  
  executer.execute(()->tv.getinstance()) at the time
* two threads will create an object that is why we will use the synchronization concept with double-check.

**Reflection API**

* we can break singleton design patterns by using reflection API
* just use className.class.getDeclereadConstructor will return the constructor and u need to use setAccesc(true)and create newInstance().
* VImp to get rid of this reflection Api inside constructor will check if the object has already been created then will return Exception this is how we get rid of reflection API

**ENUM instead of class**

* we can use an enum instead of a class and we need just any variable without type and just remove everything and use class name.instance name to get the object

**Deserialization**

* by using deserialization we can break design pattern inside somewhere conning can be done we can resolve by byreadResolve()

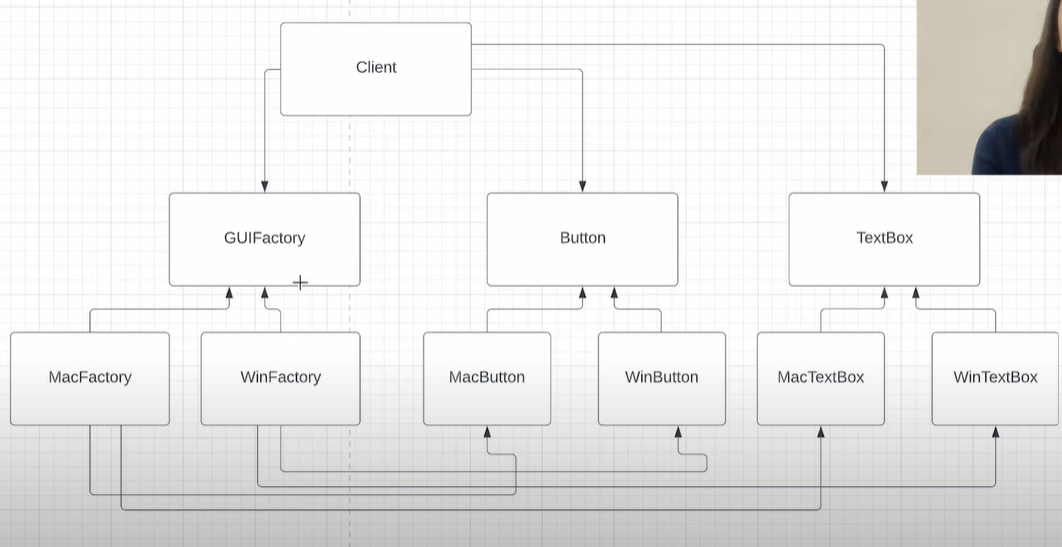
**Cloning**

* if we implement clone method from object class it will give you the new object
* so we override the clone method in our class by providing clone as our object only

**2)Factory Design Pattern**

* if we have a parent (Interface) and we are providing implementation of a parent in a child so if we are creating the object of child based on i/p it is not recommended to create an object with client class It is not good practice to show the object creation in front of the client so we will create factory Class.
* in that class, we create a static method that will return the use object based on our type i/p which object you want and we return the object to the client without knowing his internal logic for the same

**3)Abstract Design pattern**

* it provides more flexibility as it adds again new layer.
* In the factory design pattern, we have a factory which going to create abstract classes or object
* Whereas in abstract factory design patterns, we have different factories which going to create objects we have to just mention which object we want.  
    
  
* Whenever you need another level of abstraction over a group of factories we can go with abstractor factory.