Singleton pattern

The singleton is a globally available single instance object.

**Examples of swift singletons**

AppDelegate

FileManager , UserDefaults , UIApplication , and UIAccelerometer

**Advantages of a singleton**

Singleton pattern can be implemented interfaces

It can be also inherit from other classes

It can be lazy loaded  
it has status initialzation

It can be extended into a factory pattern  
it help to hide dependencies

**Disadvantages of a singleton**

1. They are generally used as a global instance, why is that so bad? Because you hide the dependencies of your application in your code, instead of exposing them through the interfaces.
2. They violate the [single responsibility principle](https://en.wikipedia.org/wiki/Single_responsibility_principle): by virtue of the fact that they control their own creation and lifecycle.
3. They inherently cause code to be tightly [coupled](https://en.wikipedia.org/wiki/Coupling_%28computer_programming%29). This makes faking them out under test rather difficult in many cases.
4. They carry state around for the lifetime of the application. Another hit to testing since you can end up with a situation where tests need to be ordered which is a big no no for unit tests. Why? Because each unit test should be independent from the other.