# Design Patterns

* They are reusable solution that will solve problems that occurs pretty frequently while coding.
* So, some people saw that this problem keeps happening across multiple coders and decided to standardize a solution every time you come across to that problem to make your life easier.
* They are known as the best practice a programmer can do to solve common problems while designing your application.

## Singleton Pattern

* This pattern revolves around creating only one concurrent instance of a class.
  + So, you can only have one object created from that class.
  + This class provides a way to let other classes have direct access to the single object.
* Advantages
  + Provides a global point of access for multiple classes.
  + It is very easy to maintain since there is only one instance class.
* Disadvantage
  + It is very difficult to unit test since it has global access.
  + Cannot perform multi-threading that well.

## Factory Pattern

* Allows the creation of objects without exposing the creation logic.
* We will use an interface to abstract the implementation from the user.
* It allows the object to be created at run-time when it is required.
* The end-user will just need to tell the factory what class it wants to get an object from.

## Dependency Injection

* A dependency is anything that an object requires to function.
* An example of a dependency is when our BL (Business Logic) requires the DL (Data Logic) to function properly.
* You just haven’t noticed but I already implemented dependency injection in our BL class and UI project.
  + I created a constructor that will take in an interface from the DL and since it is an interface, it does not have any implementation.
  + I can easily switch my DL to leverage a different way to store data (it will be a cloud database next week) and I do not have to change my BL at all.
  + That is why we use dependency injection.

# Test Driven Development (TDD)

* TDD is a software development process that creates test cases for software requirements first before developing the software.
* In the engineering world, it is the same as creating a test first before creating the product.
  + Ex: let say you want your **future product to be waterproof**. You will first create a test that either **spray water or exposed to a direct stream of water or just dump the whole thing on a pool of water**.

## The flow of TDD

1. You create a test case – what you expect the feature is suppose to do.
2. Run the test case which will obviously fail.
3. Write the code so the new test case will pass.
4. Make sure the old test cases will not fail after implementing a new feature.
5. Clean up the code and have proper documentation for other developers.

## Unit testing

* Testing small features of your code.
* Helpful when you want to make sure everything is working perfectly as intended.
* Any new features added on will also check if old features are still working.
* We will use **xunit** framework for our unit testing.

## Arrange

* This is where you initialize objects or some values you will need for the test.

## Act

* Invokes the method/function under the test with the arranged parameters defined in “Arrange”.

## Assert

* Verifies that the action of the method under the tests behaves as expected.