DESIGN PATTERNS

What are design patterns? Design patterns are reusable solutions to common programming problems that go with designing and building an application. These blueprints/solutions to application requirements are proven and tested. Design patterns make your applications reliable, scalable and easily maintainable. Design patterns are used to solve problems of object generation and integration when creating applications. They are optimized templates that can be applied to real world problems.

The original concept of design patterns was published in a book titled “Elements of reusable Object-Oriented software” and was written by “The Gang of Four” or GoF.

http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf

https://github.com/dieforfree/edsebooks/blob/master/ebooks/Design%20Patterns%2C%20Elements%20of%20Reusable%20Object-Oriented%20Software.pdf

There are three types of design patterns, these being **creational**, **structural**, **behavioral**.

Creational Design Pattern:

This type deals with object creation and initialization. This pattern gives the program flexibility in deciding which objects need to be created for a given case. Examples of this are Singleton, Factory, and Abstract Factory patterns.

GoF have identified five design patterns that belong to this category these being Singleton, Factory, Abstract Factory, Builder and Prototype Patterns.

Structural Design Pattern:

This type of pattern deals with classes and object composition. This pattern focuses on decoupling the interface and the implementation of classes and their objects. Examples of this are Adapter and Bridge.

Behavioral Design Pattern:

This type of pattern deals with communication between classes and objects. Examples of this pattern are Chain of Responsibility, Command and Interpreter.

Now On To Design Patterns:

Singleton

Used to ensure that only one instance of an object is created. Any future requests for a new instance of the object are referred to the same original instance that was created. These run on their own threads. The original object that was created can create threads/processes, but all processes come from a single object and there is only once instance of that object that was created.

The singleton pattern controls concurrent access to the resource and ensures that there is only one object available across the application. It provides global access to the instance by declaring all constructors of the class to be private and providing a static method that returns a reference to the original instance. The instance is stored as a private static variable.

-------- in progress