Singleton Pattern

THE MASTERMIND GROUP

What is a design pattern?

- In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. It is a description or template for how to solve a problem that can be used in many different situations.
- A design pattern isn't a finished design that can be transformed directly into code.
- It is a description or template for how to solve a problem that can be used in many different situations.
- Or in other words someone that has already solved your problem. It is something that you can use and there is no point of reinventing the wheel

There are three main kinds of design patterns:

- **Structural** patterns generally deal with relationships between entities, making it easier for these entities to work together.
- **Creational** patterns provide instantiation mechanisms, making it easier to create objects in a way that suits the situation. (Singelton)
- **Behavioral** patterns are used in communications between entities and make it easier and more flexible for these entities to communicate.

Singelton Design pattern

- Ensure a class has only one instance, and provide a global point of access to it.

- Singletons maintain a static reference to the sole singleton instance and return a reference to that instance from a static instance() method.

Use cases

- The logging class
- Managing a connection to a database
- File manager

Advantages vs Disadvantages

- Advantages:
  - Lazy initialization (useful if initiation is a heavy process e.g. creating a DB connection)
- Disadvantages
  - Makes it difficult to apply unit testing as you have introduced global state
  - If the Singleton class is loaded by 2 different class loaders we'll have 2 different classes, one for each class loader

Multithreading

- Create a version of the singleton that is thread safe
- Commit it to github/svn


QUESTIONS?
COMMENTS?