**Singleton Design Pattern:**

**What ?**

Design pattern are reusable solutions to commonly occuring problems in software design. “Singleton design pattern is a pattern that restrict the instantation of a class to one object.”

**When ?**

1. Ensure only one instance of a class is been created.

2. Provide global point of access to a object.

**Why ?**

When there must be exactly one instance of a class, and when it must be accessible to clients from global access point.

Eg. Logger class, Configurator file reader class

**How to execute ?**

*Program:*

#include<iostream>

class Singleton

{

private:

static Singleton\* instance;

Singleton(){}

public:

static Singleton\* getInstance(){

if(instance == NULL){

instance = new Singleton();

std::cout << "Instance is created\n";

} else {

std::cout << "New Instance cannot be created\n";

}

}

};

Singleton\* Singleton::instance = NULL;

int main()

{

Singleton\* obj = Singleton::getInstance();

Singleton\* obj1 = Singleton::getInstance();

return 0;

}

*Output:*

*Instance is created*

*New Instance cannot be created*

**Diasadvantages:**

Its not thread safe.