Singleton

Tuesday, June 26, 2018 9:55 PM

Singleton Concept:

1. Singleton is a design pattern that restricts the instantiation of class to just one object

2. Implementation:
   a. Ensure that only 1 instance of the singleton class ever exists
   b. Provide global access to that instance, usually for global management

3. Methodology:
   a. Declaring all constructors of the class as private
   b. Providing a static method that returns a reference to the instance of the object

4. Code Implementations:

Implementation:

5.

```cpp
#include <iostream>

using namespace std;

class Singleton
{
    private:
        /* Here will be the instance stored. */
        static Singleton* instance;

        /* Private constructor to prevent instancing. */
        Singleton(){ cout << "Constructor Called"<< endl; };
        //~Singleton(){cout << "Destrcutor called" << endl;};

    public:
        /* Static access method. */
        static Singleton* getInstance() {


            if (instance == nullptr)
            {
                cout << "Creating Object for first time" << endl;
                instance = new Singleton();
            }

            else cout << "Object already created" << endl;


            return instance;

        };

};

/* Null, because instance will be initialized on demand. */
Singleton* Singleton::instance = nullptr;

```

Usage:

```cpp
int main()
{
    //new Singleton(); // Won't work
    //Singleton *t = new Singleton();
    Singleton* s = Singleton::getInstance(); // Ok
    Singleton* r = Singleton::getInstance();


    /* The addresses will be the same. */
    cout << "Pointer to first object  = " << s << std::endl;
    cout << "Pointer to second object  = " << r << std::endl;

}

```

Output:

```
$g++ -std=c++11 -o main *.cpp
$main
Creating Object for first time
Constructor Called
Object already created
Pointer to first object  = 0x68bc30
Pointer to second object  = 0x68bc30
```

6. **IMP Notes:**
   a. Default Constructor is made private. Therefore, now, if a Singleton object is requested to be made outside of the class, it will give COMPILER error
   b. Instance is a static class member. Therefore, only 1 copy will be maintained for the whole class
   c. Because, we cannot initialize a static class member inside the class, therefore it is initialized outside of the class
   d. getInstance is a static class member function, and hence, can be declared and defined inside the class itself. This is different for static data member of the class.