1. Signleton class

- one object of singleton class

- thread safe - synchronization

- double check locking - how can you use it in singleton?

*In object-oriented programming, a singleton class is a class that can have only one object at a time.  
After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. So whatever modifications we do to any variable inside the class through any instance, it affects the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.  
To design a singleton class:*

1. *Make constructor as private.*
2. *Write a static method that has return type object of this singleton class. Here, the concept of*[*Lazy initialization*](https://en.wikipedia.org/wiki/Lazy_initialization)*is used to write this static method.*

class Singleton  
{  
 private static Singleton *single\_instance* = null;  
 // variable of type String  
 public String s;  
 // private constructor restricted to this class itself  
 private Singleton()  
 {  
 s = "object of singleton class";  
 }  
 // static method to create instance of Singleton class  
 public static Singleton getInstance()  
 {  
 if (*single\_instance* == null)  
 *single\_instance* = new Singleton();  
  
 return *single\_instance*;  
 }  
}  
// Driver Class  
public class Main  
{  
 public static void main(String args[])  
 {  
  
 Singleton obj1 = Singleton.*getInstance*();  
 Singleton obj2 = Singleton.*getInstance*();  
 Singleton obj3 = Singleton.*getInstance*();

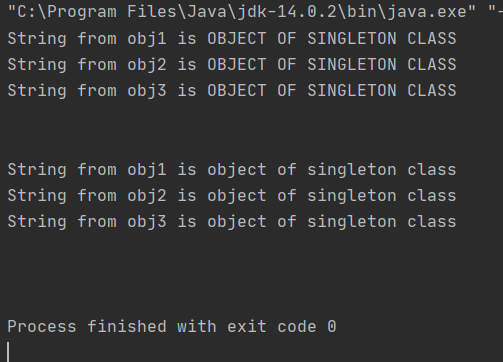
obj1.s = (obj1.s).toUpperCase();

System.*out*.println("String from obj1 is " + obj1.s);  
 System.*out*.println("String from obj2 is " + obj2.s);  
 System.*out*.println("String from obj3 is " + obj3.s);  
 System.*out*.println("\n");

obj3.s = (obj3.s).toLowerCase();

System.*out*.println("String from obj1 is " + obj1.s);  
 System.*out*.println("String from obj2 is " + obj2.s);  
 System.*out*.println("String from obj3 is " + obj3.s);  
 System.*out*.println("\n");  
 }  
}

Output:



*In double-checked locking, code checks for an existing instance of* [*Singleton class*](http://javarevisited.blogspot.sg/2013/03/difference-between-singleton-pattern-vs-static-class-java.html)*twice with and without locking to double ensure that no more than one instance of singleton gets created.*

class Singleton {  
 private volatile static Singleton *\_instance*;  
 public String s;  
 private Singleton() {  
 // preventing Singleton object instantiation from outside  
 s="string set in the class";  
 }  
 //double locking  
 public static Singleton getInstance() { if (*\_instance* == null) {  
 synchronized (Singleton.class) {  
 if (*\_instance* == null) {  
 *\_instance* = new Singleton(); }  
 }  
 }  
 return *\_instance*;  
 }  
}  
  
public class Main  
{  
 public static void main(String args[])  
 {  
  
 Singleton obj1 = Singleton.*getInstance*();  
 Singleton obj2 = Singleton.*getInstance*();  
 Singleton obj3 = Singleton.*getInstance*();  
  
 obj1.s = (obj1.s).toUpperCase();  
  
 System.*out*.println("String from obj1 is " + obj1.s);  
 System.*out*.println("String from obj2 is " + obj2.s);  
 System.*out*.println("String from obj3 is " + obj3.s);  
 System.*out*.println("\n");  
  
 obj3.s = (obj3.s).toLowerCase();  
  
 System.*out*.println("String from obj1 is " + obj1.s);  
 System.*out*.println("String from obj2 is " + obj2.s);  
 System.*out*.println("String from obj3 is " + obj3.s);  
 System.*out*.println("\n");  
 }  
}