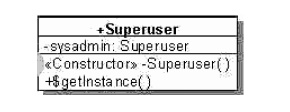
**Singleton**

       The singleton pattern deals with situations where only one instance of a class must be created. Take the case of a system administrator or superuser. This person has the right to do everything in a computer system. In addition we will also have classes representing normal users. Therefore we must ensure that these classes have no access to the super user constructor. The solution to this problem in C++ and Java is to declare the superuser constructor private.

 The super user class itself has a private static attribute sysadmin, which is initialised using the class constructor. Now we get an instance of the super user class with a public static method that returns sysadmin. Here is the class diagram:



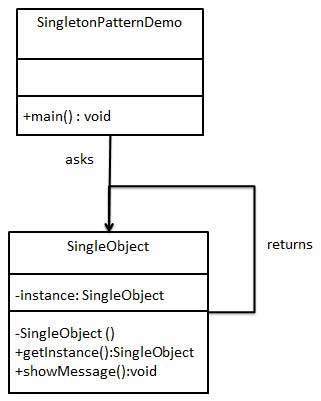
Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

**Implementation**

We're going to create a *SingleObject* class. *SingleObject* class have its constructor as private and have a static instance of itself.

*SingleObject* class provides a static method to get its static instance to outside world. *SingletonPatternDemo*, our demo class will use *SingleObject* class to get a *SingleObject* object.



Step 1

Create a Singleton Class.

*SingleObject.java*

public class SingleObject {

//create an object of SingleObject

private static SingleObject instance = new SingleObject();

//make the constructor private so that this class cannot be

//instantiated

private SingleObject(){}

//Get the only object available

public static SingleObject getInstance(){

return instance;

}

public void showMessage(){

System.out.println("Hello World!");

}

}

Step 2

Get the only object from the singleton class.

*SingletonPatternDemo.java*

public class SingletonPatternDemo {

public static void main(String[] args) {

//illegal construct

//Compile Time Error: The constructor SingleObject() is not visible

//SingleObject object = new SingleObject();

//Get the only object available

SingleObject object = SingleObject.getInstance();

//show the message

object.showMessage();

}

}

Step 3

Verify the output.

Hello World!