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 way to create an object.

This pattern involves a single class which is responsible to creates own object while making sure that only single object get 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.

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();

}

}

Singleton in case of multithreaded programs:

<http://howtodoinjava.com/2012/10/22/singleton-design-pattern-in-java/>

[view source](http://howtodoinjava.com/2012/10/22/singleton-design-pattern-in-java/#viewSource)[print](http://howtodoinjava.com/2012/10/22/singleton-design-pattern-in-java/#printSource)[?](http://howtodoinjava.com/2012/10/22/singleton-design-pattern-in-java/#about)

01.public class EagerSingleton {

02.    private static volatile EagerSingleton instance = null;

03.

04.    // private constructor

05.    private EagerSingleton() {

06.    }

07.

08.    public static EagerSingleton getInstance() {

09.        if (instance == null) {

10.            synchronized (EagerSingleton.class) {

11.                // Double check

12.                if (instance == null) {

13.                    instance = new EagerSingleton();

14.                }

15.            }

16.        }

17.        return instance;

18.    }

19.}

Above code is the correct implementation of singleton pattern.

EXAMPLES IN EXISTING JAVA

1. Singleton: java.lang.System

### [Singleton](http://en.wikipedia.org/wiki/Singleton_pattern) (recognizeable by creational methods returning thesameinstance (usually of itself) everytime)

* [java.lang.Runtime.getRuntime()](http://docs.oracle.com/javase/6/docs/api/java/lang/Runtime.html#getRuntime%28%29)
* [java.awt.Desktop.getDesktop()](http://docs.oracle.com/javase/6/docs/api/java/awt/Desktop.html#getDesktop%28%29)

Every Java application has a single instance of class **Runtime** that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the getRuntime method.