3 ways to break singleton

1) cloning

2) reflection

3) serialization-->

**Cloning**

Again this can be avoided if we don’t implement Cloneable and override clone method. But the best practice is to

1) Return same instance under clone method

2) As usual return super.clone() inside clone() method. This will not affect anything. Because Static values will not be cloned.

Note: cloneInstance.hashCode() will return different value. But cloneInstance.getInstance().hashCode() will return same hashcode value.

3) Throw CloningNotSupportedException under clone() method.

**Reflection**

* Through Reflection Load class
* modify private constructor to public
* create instance.

**How to avoid Reflection to create another instance of Singleton**

public class Singleton {

private static final Singleton INSTANCE = new Singleton();

private Singleton() {

// Check if we already have an instance

if (INSTANCE != null) {

throw new IllegalStateException("Singleton" +

" instance already created.");

}

}

public static final Singleton getInstance() {

return INSTANCE;

}

private Object readResolve() throws ObjectStreamException {

return INSTANCE;

}

private Object writeReplace() throws ObjectStreamException {

return INSTANCE;

}

public Object clone() throws CloneNotSupportedException {

// return INSTANCE

throw new CloneNotSupportedException();

}

}

**Serialization:**

Sometimes we need singleton object to be serialized. But if you deserialize in the same class you will get different instance which will violate singleton principle. Override readResolve() method which will call be called during de-serialization.

protected Object readResolve() {

    return getInstance();

}

SerializedSingleton instanceTwo = (SerializedSingleton) in.readObject();

in.readObect() will call readResolve() rather loading the values from file and assign it to left side.

SerializedSingleton instanceOne = SerializedSingleton.getInstance();

        ObjectOutput out = new ObjectOutputStream(new FileOutputStream(

                "filename.ser"));

        out.writeObject(instanceOne);

        out.close();

        //deserailize from file to object

        ObjectInput in = new ObjectInputStream(new FileInputStream(

                "filename.ser"));

        SerializedSingleton instanceTwo = (SerializedSingleton) in.readObject();

        in.close();

        System.out.println("instanceOne hashCode="+instanceOne.hashCode());

        System.out.println("instanceTwo hashCode="+instanceTwo.hashCode());