

Singleton DP

4-Feb-20 11:01 PM

91

Introduction

- Sometimes we need only one object of a particular type e.g.
 - Scanner for keyboard
 - Registry
 - Logger
 - Mouse device
 - Printer spooler
- How to ensure that one and only one object is created for a particular class in our application?



Solution (Java)

```
public final class MyClass {  
    private static MyClass ins =  
        new MyClass();  
    private MyClass(){}  
    public static MyClass getMyClass() {  
        return ins;  
    }  
}
```



4-Feb-20 11:01 PM

93

This pattern ensures that a class has at the most one instance.

This pattern is in widespread use.

It is usually used along with other patterns e.g. Abstract Factory, Builder, Prototype, Facade, etc.

Singleton code in C# present with participants.

Solution (Java)

```
public enum MyClass {  
    INSTANCE;  
    //... function of the class  
}
```

4-Feb-20 11:01 PM

94

This pattern ensures that a class has at the most one instance.

This pattern is in widespread use.

It is usually used along with other patterns e.g. Abstract Factory, Builder, Prototype, Facade, etc.

Solution (C++)

```
class MyClass {
    static MyClass *ins;
    MyClass() {}
public: static MyClass *getMyClass() {
    //Not thread-safe.
    if (ins == 0) ins = new MyClass();
    return ins;
}
};
MyClass *MyClass::getMyClass;
```



4-Feb-20 11:01 PM

95

Thread safe Singleton code in C++11 with participants. For older C++ version use locking; code present with boost locking with participants.

Double-Locked checking

```
private static volatile
    MyObj inst = null;
public static MyObj getInst() {
    if (inst == null)
        synchronized (MyObj.class) {
            if(inst == null)
                inst = new MyObj();
        }
    return inst;
}
```

Find the bug in this Singleton?

4-Feb-20 11:01 PM

96

<http://java67.blogspot.in/2015/09/thread-safe-singleton-in-java-using-double-checked-locking-pattern.html>

//The code is buggy because JVM can reorder statements within the synchronized block. The bug is present only before Java 1.5

//So other thread may get object which is not initialized.

Right code:

```
private static MyObj inst = null;
public static synchronized MyObj getInst() {
    if (inst == null) inst = new MyObj();
    return inst;
}
```

Non-Software Analog

- A radio station broadcasts at a specific frequency, and within a given context (a "market").
 - No other station can broadcast at this same frequency.

Singleton Registry

- A application with multiple Singletons can maintain a registry of Singletons in a single place.

4-Feb-20 11:01 PM

98

The registry maintains a map between string names and Singletons.

We request for a singleton by its name.

If the singleton is present, the registry returns it.

If the singleton has not been created, it is created and then henceforth returned.

Misuse of Singletons

- Singletons are not same as Global Variables.

4-Feb-20 11:01 PM

99

Singleton's should be used to ensure that the object has only one instance and *not* to get that instance from everywhere. E.g.

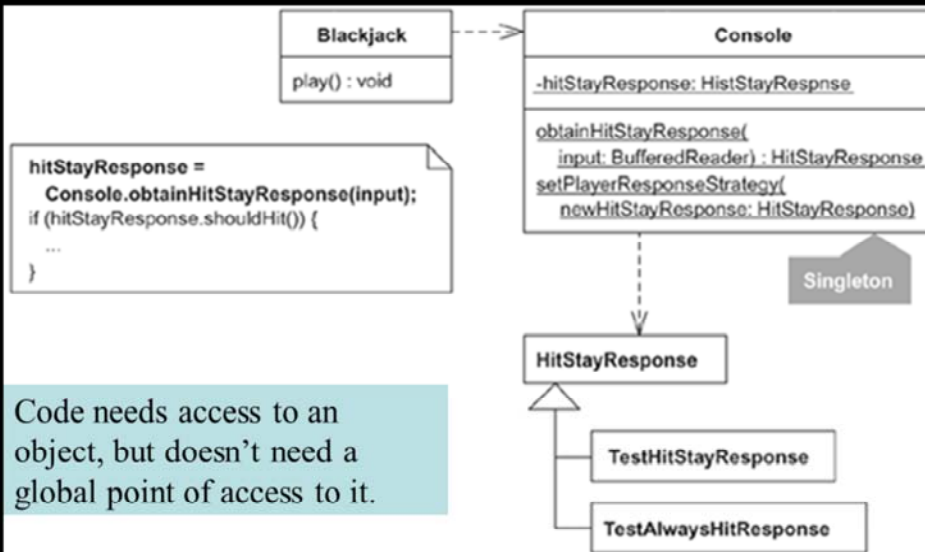
One group had a bad habit of using global data, so they did a study group on Singleton.

The next thing we know is that Singletons appeared everywhere and none of the problems related to global data went away.

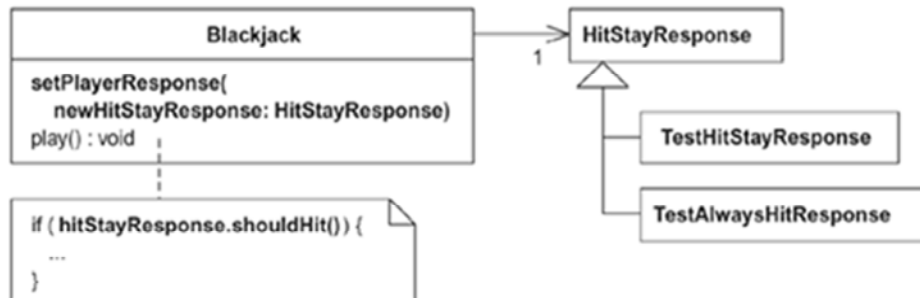
The answer to the global data question is not, "Make it a Singleton." The answer is, "Why are we using global data?"

Changing the name or syntax doesn't change the problem.

Remove Singleton



No Singleton



Move the Singleton's features to a class that stores and provides access to the object. Delete the Singleton.

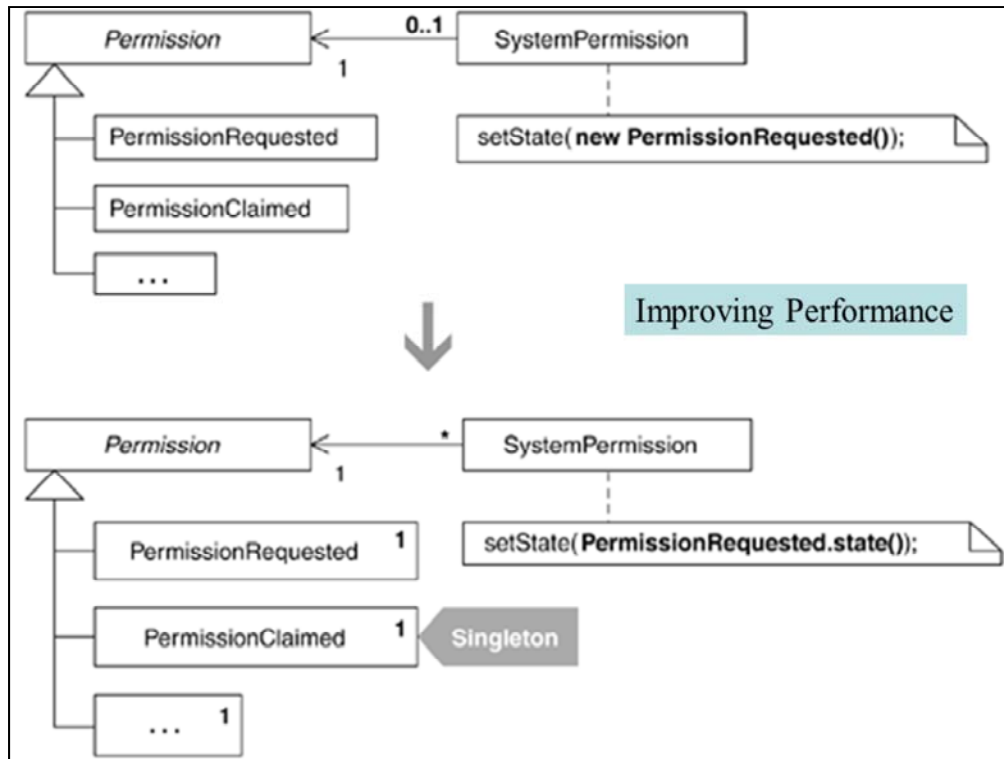
4-Feb-20 11:01 PM

101

When is a Singleton unnecessary?

Short answer: Most of the time.

Long answer: A Singleton is unnecessary when it's simpler to pass an object resource as a reference to the objects that need it, rather than letting objects access the resource globally. A Singleton is unnecessary when it's used to obtain insignificant memory or performance improvements. A Singleton isn't necessary when code deep down in a layer of a system needs to access a resource but the code doesn't belong in that layer to begin with. I could go on. The point is that Singletons aren't necessary when you can design or redesign to avoid using them.



Your code creates multiple instances of an object, and that uses too much memory or slows system performance.

Replace the multiple instances with a Singleton.

Sometimes it's a good decision to refactor to a Singleton, as in the following scenario.

- Users of your system are complaining about system performance.
- Your profiler tells you that you can improve performance by not instantiating certain objects over and over again.
- The objects you want to share have no state or contain state that is sharable.

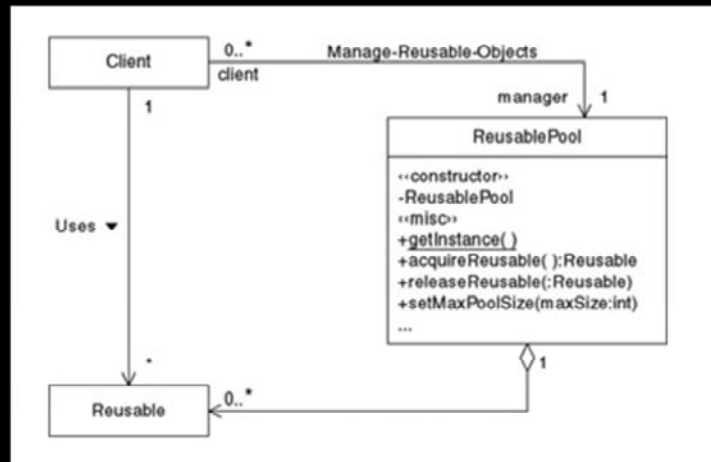
Benefits and Liabilities

- + Improves performance.
- Is easily accessible from anywhere. In many cases, this may indicate a design flaw
- Is not useful when an object has state that can't be shared.



Object Pool

- A variation of Singleton is the Object Pool pattern



4-Feb-20 11:01 PM

103

A set of objects exists e.g.

A fixed number of database connections. Threads acquire and release instances.

The ResourcePool to allocate and free connections is a singleton.

Java and c++ solution present.

Problems with Singleton

- If we serialize a singleton and then deserialize it twice, we get two objects of the Singleton class
- Deleting a Singleton instance i.e. Destroying a singleton object, is a non-trivial design problem
- A singleton is difficult to subclass
- J2EE can use multiple JVMs for EJBs. So it is difficult to implement the singleton pattern

4-Feb-20 11:01 PM

104

1) If we serialize a singleton and then deserialize it twice, we get two objects of the Singleton class. To prevent this problem

In Java: Mark all attributes as transient and have the readResolve function defined in the class.

In C#: Don't mark the class with [Serializable] attribute.

2) By using reflection, the caller can create multiple objects of a singleton class in Java. If this is a problem, in the constructor check whether an instance has been created. If yes, throw an exception.

3) Another way to implement a Singleton in Java is

```
public enum Elvis {  
    INSTANCE;  
    public void leaveTheBuilding() { ... }  
}
```

This is automatically safe from Serialization and Reflection attacks.

Assignment: MultiTon pattern

- Similar to the singleton.
- The multiton pattern expands on the singleton concept to manage a map of named instances as key-value pairs.
 - Rather than have a single instance per application (e.g. the `java.lang.Runtime` object in the Java programming language) the multiton pattern instead ensures a single instance per key.
 - http://en.wikipedia.org/wiki/Multiton_pattern
- Assignment: Write Thread-Safe MultiTon pattern code.



4-Feb-20 11:01 PM

105

Java code in Patterns project SingletonMultiton.java

C# code is OOADandPatterns solution with same name.