Freshers: Get IT JOBs Faster!

Get Grip On OOP Using Java!

C/C++ Interview Questions

C

C++ Basic

C++ Advanced

Code for C++ delete singleton instance

October 26, 2014 by Rakesh Singh

(Last Updated On: August 21, 2020)

Answer: To delete singleton instance or object in C++ programs, we need to follow a proper design to delete singleton object in the class . Especially, when multiple users are using object of singleton class in C++ program . One of the bad situations we can think that what if one user has deleted singleton object while other multiple users are still using the single instance.

Author

Rakesh Singh is a highly experienced IT professional & International Published Author…Read more

C# Programming

C# program exercises

Java Interview Questions

Core Java

Java Collections

Java Multithreading

Java Notes

Java Tutorial

Multiple Choice Questions

MCQ-C++

MCQ-Java

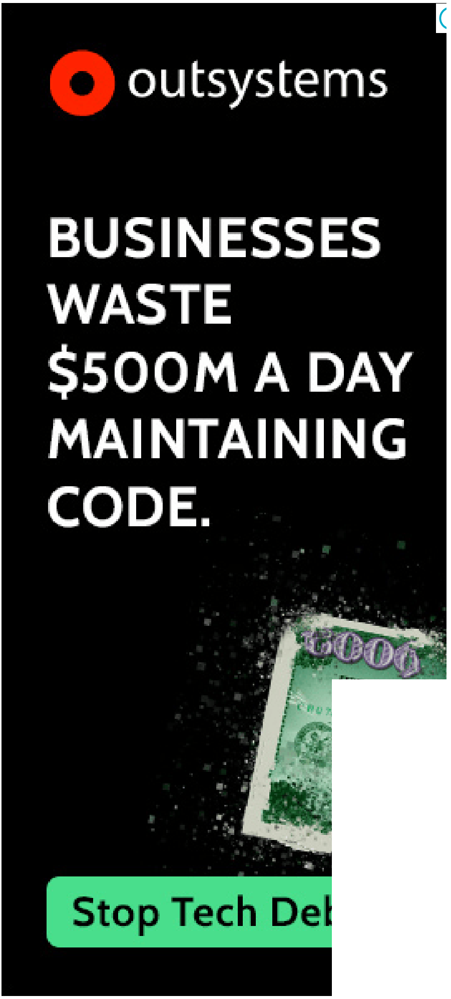
MCQ-SQL

MCQs Algorithms and Data Structure

So, one of the designs we can follow is “Reference counting” mechanism to delete singleton instance of a class in C++ program. Whenever, a user requests an instance to single class, we can increase the reference count and when he delete it we can decrease the reference count.

And deletion of singleton class object would be allow only when the count is zero.

To design C++ delete singleton instance, first, we need to make the singleton class destructor private, so, it can not be accessed from outside of the class. Hence, user cannot delete the singleton instance using the keyword “**delete”.** Also, we have to introduce a static method in the singleton class, say



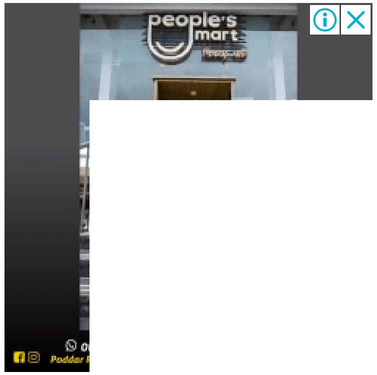
Recent Posts

Extract Numbers from String – Java

Networking General

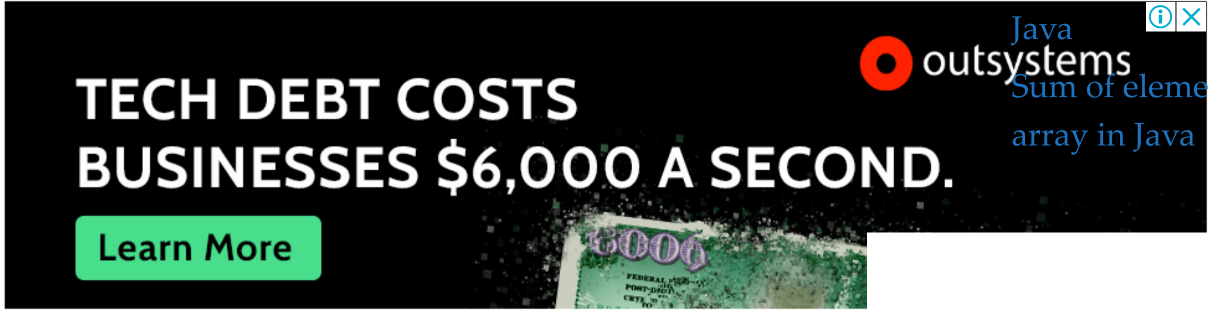
UML Diagram

Mathematical Puzzles



"releaseInstance() that will be used to delete singleton class pointer.

Introduce a new static variable "count" that will be used to track users. And, also, have two other private method "addRef()" and "releasRef()" to increase and decrease counts of the instance.



Whenever, releaseInstance() function is called, this function will check if singleton class object destruction is required or not.

C++ delete singleton instance design

C++ Code Example

Test Your Skills in OOPs Concepts with JAVA and Master Quickly

FRESHER: Learn what Level of OOPs Concepts Separates You From The Crowd in Job Interviews

Arraylist Of Objects In Java

Sum of elements in an array in Java

```
class Singleton{
public:
    static Singleton* GetInstance()
        {
                if(NULL == s_instance)
                        {
                                s_instance=new
Singleton();
                        }
                addRef();// track users

                return s_instance;
        }

        static void releaseInstance()
        {
                releasRef();

                if((0 == counter)&& (NULL !=
s_instance))
                {
                        delete s_instance;
                        s_instance = NULL;
                }
        }


private:
        // Make the constructor private so,
user can not create
        //object directly but with
GetInstance() method.
        Singleton(){
                cout<<"constructor...\n";
        }

        // don't allow users to apply delete
on objects.
        //as we will force users to call
destroyInstance method.
```

```
        ~Singleton(){
                cout<<"destructor...\n";
        }


        static void addRef(){
                ++counter;
        }
        static void releasRef(){
                --counter;
        }

    static Singleton* s_instance;
        static int counter;

};

//initialize static variables here
Singleton *Singleton::s_instance = 0;
int Singleton::counter = 0;
```

Note that this question can also be asked as to provide c++ singleton destructor example. Hence, the answer should be same as above that we need to make custom design to delete instance of singleton class.

User Sneha’s Question

Can the delete instance functionality cause infinite loop?

Author's Answer:

Before answering your question let me explain some insight of the given design in mult-threaded environment.

First, the given design is for understanding purpose and does not support multithreaded environment. In single threaded environment it does not cause any issue.

If you use the same code in the multi-threaded environment, following issue can occur.

1- inside the function GetInstance(), more than one thread can enter and can create more than one objects, in addition to that the function addRef() can be called multiple times and the value of the variable “counter” will be incremented multiple times, which is not expected.

2- In the releaseInstance function more than one thread can enter and execute the releaseRef function resulting multiple decrements of the counter variable. Which is also not expected.

So, ultimate result is that the value of the counter variable can be unpredictable and can never become ZERO in a multi-threaded environment.

If you have created hundred of objects and since, the counter is not becoming zero, the memory leak will happen. In other word, the releaseInstance() function will not be able to delete any instance as it will not enter the IF block.

So, coming to your question.

In multi-threaded environment if you have an application that continuously monitor the value of “counter” variable to become zero, and when the counter value becomes zero, then your application takes some kind of action. If the value of the counter never becomes zero, obviously, your application will fall in an infinite loop.

Solution to get the design into a multi-threaded environment

To solve the problem, you can use locking an unlocking mechanism inside the GetInstance() and release instance() method. See the commented LOCK and UNLOCK.

```
static Singleton* GetInstance() {

        //LOCK
        if(NULL == s_instance) {

                s_instance=new Singleton();
        }

        addRef();// track users

        // UNLOCK

        return s_instance;
}



static void releaseInstance() {

        //LOCK
        releasRef();
        if((0 == counter)&& (NULL !=
s_instance)) {
                delete s_instance; s_instance
= NULL;
        }
        //UNLOCK
}
```

Please look here the complete detail and code of a Singleton class in C++ in multi-threaded environment.

Related Posts:

- Explain C# Singleton class with code example - Simple Steps
- How to delete array of objects in C++? Proof by C++…
- Before Delete and After Delete Triggers
- Does each thread have its own copy of JVM instance?
- Show C# Static and Instance methods call with simple program

C++ Advanced

delete singleton object c++, Singleton

‹ Design Singleton class for multi-threaded environment in C++

› What are issues if we mix new and free in C++?

4 thoughts on “Code for C++ delete singleton instance”

Sneha
August 20, 2020 at 1:53 pm

delete instance can cause infinite loop as per my other reading is it true or not.

Reply

Rakesh Singh
August 21, 2020 at 6:49 pm

Hey Sneha, this is a fantastic question. I have updated the answer

to your question in the bottom of the post with your name.
Thanks! Please let me know if you have further question.

Reply

eshwar
April 24, 2020 at 11:59 pm

I have not seen this implementation else where. is this thread safe

Reply

Rakesh Singh
April 27, 2020 at 4:51 am

Thanks for the questions eshwar. No, this is not thread safe. Please consider reading thread safe singleton class here.

Reply

Leave a Comment

Name *

Email *

Website

Math Captcha

□ − 1 = 1

Post Comment

Get Grip On OOP Using Java!

Get Interview Calls & JOBS Faster!

ources

nail!

SUBSCRIBE

Please do not forget to click on the activation link

that will be sent to your email Id.

private policy