Lecture 10: Singleton Design Pattern

Page No.
Date

Introduction

- Singleton is a type of class where class restricts itself to create only one instance/ object and agar hum dusra object banaye toh bhi same instance return kare.

Understand object creation under hood

A

- Humme suppose ek class ka object create krna hai - `A * a = new A();`

⇓

What happens when this stmt gets executed

⇒ We have two types of memory stack & Heap

Stack ⇓ stores primitive datatype like int, char, bool

Heap ⇓ non-primitive datatype like custom classes ke obj, func, etc.

- Ab hum jab `new A()` likhege toh ek object of class A banega.

A() → this calls constructor of class. [constructor is called by itself whenever we call/ create object of that class]

→ constructor humare object ko initiate karta hai yaane default values deta hai

→ Aur agar user ne constructor nhi banaya toh CPP/Java default constructor bnti hai joh user ko

- What happens : A * a = new A()

1. Jaise hi new A() is called sbse pehle mem di jayegi class ko depending on class size.

2. Class A ka constructor call hoga - agar user ne nhi diya hai toh default constructor call ho jayega.

3. Ek variable a* hum stack mein rkhege & woh humare heap address ko point krega.

How to stop multiple object creation of class

code in vs bss short description here

Method 1 :

1. Hum constructor ko private kar dege taaki constructor class ke bahar se call na ho.

2. Hum jb yeh krege toh pehla object bhi nhi banega toh ab hum ek static variable and get method use krege joh publicly access honga.

3. Hum static variable - instance ko nullptr initially assign krege (outside main & class)

datatype

4. Fir ek static singleton * getInstance naam ka method baneyge jha hum check krege agr instance == nullptr ka toh ~~aur~~ neeo object create krdo ~~wor~~ and instance mein save karo / agar != nullptr ke toh instance return krdo.

→ But this method will fail, agar humara application multithreading ko support krta hoga - yaane at same multiple threads saath mein access kre toh shyd multiple instance create ho jaye.

Method 2: Making class Thread safe.

1. Ab hum yaha lock use krege - locks mtlb security check agar function ko lock hai toh aap/threads andar nhi jaa skte

2. Lock it using : lock-gaurd <mutex> lock(mtx);

3. Ab thread jb lock leave karegi tab hi dusri thread aa skti.

4. And lock humme tab hi lagana hai jab nullptr ho instance agar nhi hai toh koi issue nhi --- Aisa isliye kyuki lock operation expensive hai

- still yha problem hai suppose T_1 & T_2 do threads hai dono saathmein aaye and unhone dono ne if condition validate krli i.e. (instance == nullptr) agar or yaha par one ke baad suppose T2 ne lock lagaya & access le liya our instance bana diya toh T_1 thread jinse if condn validate nhi krti woh ~~same~~ nxt stmt yaane lock wali stmt se continue kregi which is wrong

Page No.
Date

Solution : Hum double check lagayege i.e. lock lagane ke baad bhi hum check krege if condition again

→ Ek our solution hai jismein hummein locks ki jarrorat hi nhi naa hi if condition ki

Method 3 : Eager Initialization

1. Create object of the instance even before running our main method - using static var

2. And jab bhi main method mein jb bhi object creation ke liye call jayega toh woh instance ko bona dena.

Problems in this method.

1. Suppose main method mein humne kbhi object hi nahi banaya toh?
It will lead to huge memory kyuki humari class kbhi kbhi bahoot badi hoti hai & jyada memory bhi leti hai & agar use hi nhi kiya toh memory loss

2. Solution : Hum Eager Initifization tabhi use kare jab hum sure ho ki hum object ko call karege.

Overall Method working
- ① Create a private constructor
- ② Create a static instance (getInstance) that returns the same instance every time.

Real-World use case.

① Logging system
② Database Connection
③ Configuration Manager
} details video timestamp - 29:17

Where not to use singleton

1. Jaha hume pata hai ki single instance se kaam nhi chlega we will need multiple instance jaise ki online games jha team banakr play krte.

2. Thread safety important hai : multithreading code likhna hard hai toh avoid kro singleton jb tk ap sure nhi ho ki yeh bahoot jyada important hai - kyuki yaha unit test case bhi banana hard hota hai