Статичні члени та методи класу. Сінглетон

Статичні члени класу

Існує тільки ***одна копія статичного члена***. ***Статичний член є спільним*** для ***всіх об'єктів*** класу.

```
#include <iostream>
using namespace std;
class Box {
  public:
    static int objectCount; // статичний член – лічильник обєктів
    // Конструктор
    Box(double l = 2.0, double b = 2.0, double h = 2.0) {
      cout <<"Constructor called." << endl;
      length = l;
      breadth = b;
      height = h;
      // збільшуємо на одиницю лічильник об’єктів
      objectCount++;
    }
    double Volume() {
      return length * breadth * height;
    }
```

Статичні члени класу

```
private:

    double length;     // Length of a box
    double breadth;    // Breadth of a box
    double height;     // Height of a box
};

// Ініціалізуємо статичний член класу
int Box::objectCount = 0;

int main(void) {
  Box Box1(3.3, 1.2, 1.5);
  Box Box2(8.5, 6.0, 2.0);
  // Друкуємо загальну кількість обєктів
  cout << "Total objects: " << Box::objectCount << endl;
  return 0;
}
```

Результат:

Constructor called.
Constructor called.
Total objects: 2

Статичні методи класу

Статичний метод може отримати ***доступ тільки до статичного члена класу***, *інших статичних методів* і будь-яких інших функцій поза класом.

```
#include <iostream>
 using namespace std;
class Box {
   public:
      static int objectCount;
***
static int getCount() { // статичний метод для виведення лічильника
        return objectCount;
     }
****
// Ініціалізація статичного члену пітрібна в будь-якому файлі що використовує лічильник
int Box::objectCount = 0;
int main(void) {
   // Друкуємо загальну кількість обєктів
   cout << "Inital Stage Count: " << Box::getCount() << endl;

   Box Box1(3.3, 1.2, 1.5);
   Box Box2(8.5, 6.0, 2.0);
// Друкуємо загальну кількість обєктів
   cout << "Final Stage Count: " << Box::getCount() << endl;
   return 0;
}
```

Результат:
Inital Stage Count: 0
Constructor called.
Constructor called.
Final Stage Count: 2

Сінглетон (Singleton)

```
//File: logger.hpp
#include <string>
class Logger{
public:
   static Logger* Instance(); // створення класу (замість конструктору)
   bool openLogFile(std::string logFile); // читання класу з файлу
   void writeToLogFile(); // запис у файл
   bool closeLogFile(); // закриття файлу
private:
   Logger(){};  // Приватний (Private) тому не може бути викликаний
   Logger(Logger const&){};             // copy constructor  -  теж приватний
   Logger& operator=(Logger const&){};  // присвоєння (assignment)  – теж приватний
   static Logger* m_pInstance;  // це лічильник класу
};
```

Реалізація та використання сінглетону

```
//File: logger.cpp
#include <stddef.h>  // defines NULL
#include "logger.h"
// Глобальна статична змінна зберігає обєкт та гарантує єдиність екземпляру
Logger* Logger::m_pInstance = NULL;
/** Цей метод потрібен для ініціалізації класу бо виклик конструктору заборонений.*/
Logger* Logger::Instance(){
  if (!m_pInstance)   // Лише один екземпляр класу дозволений
    m_pInstance = new Logger;
  return m_pInstance;
}
```

```
bool Logger::openLogFile(std::string _logFile)
{
***// код для читання з файлу
}
/*  Реалізація методів writeToLogFile() та closeLogFile()
*/

***
 // Це може бути будь-який файл, що включає logger.hpp

/// Використання сінглетону
Logger::Instance()->openLogFile("logFile.txt");
// або
Logger * ptr = Logger::Instance();
ptr -> openLogFile("logFile.txt");  //  але лише один з цих варіантів
ptr -> writeToLogFile; ptr-> closeLogFile();
```