- Examples

Let’s discuss the examples of thread-safe initialization of data in this lesson.

WE'LL COVER THE FOLLOWING

Example 1

The short example demonstrates the application of `std::call_once` and the `std::once_flag`. Both are declared in the header `<mutex>`.

```cpp
// safeInitializationCallOnce.cpp

#include <iostream>
#include <thread>
#include <mutex>

std::once_flag onceFlag;

void do_once(){
  std::call_once(onceFlag, [](){ std::cout << "Only once." << std::endl; });
}

int main(){

  std::cout << std::endl;

  std::thread t1(do_once);
  std::thread t2(do_once);
  std::thread t3(do_once);
  std::thread t4(do_once);

  t1.join();
  t2.join();
  t3.join();
  t4.join();
  std::cout << std::endl;
}
```

Explanation

- The program starts four threads (lines 17 - 20). Each of them invokes `do_once`. The string “only once” is, as a result, displayed only once.
- The famous singleton pattern guarantees that only one instance of an object will be created. This is a challenging task in multithreading environments. Due to `std::call_once` and `std::once_flag`, the job is made much easier.

Example 2

Here is an example of the thread-safe Meyers Singleton pattern:

```
// safeInitializationStatic.cpp

#include <iostream>

class MeyersSingleton{

  private:

    MeyersSingleton()= default;
    ~MeyersSingleton()= default;

  public:

    MeyersSingleton(const MeyersSingleton&)= delete;
    MeyersSingleton& operator=(const MeyersSingleton&)= delete;

    static MeyersSingleton& getInstance(){
      static MeyersSingleton instance;
      return instance;
    }
};


int main(){

  std::cout << std::endl;

  std::cout << "&MeyersSingleton::getInstance(): "<< &MeyersSingleton::getInstance() << std::
  std::cout << "&MeyersSingleton::getInstance(): "<< &MeyersSingleton::getInstance() << std::

  std::cout << std::endl;

}
```

```
}
```

Explanation

- By using the keyword `default`, you can request special methods from the compiler. These methods are special because they are created by the compiler.

- `delete` results in the following: the automatically generated methods (constructor, for example) from the compiler will not be created and cannot be called. They will generate a compile time error.

What's the point of the Meyers Singleton in multithreading programs? The Meyers Singleton is thread-safe.

Know your Compiler support for static

If you use the Meyers Singleton pattern in a concurrent environment, be sure that your compiler implements static variables with the C++11 thread-safe semantic. Programmers often rely on the C++11 semantics of static variables, but their compiler does not support this function. This may result in the creation of more than one instance of a singleton.

To learn more about Thread-Safe Initialization of a Singleton, check this.

Level up your understanding of this topic with an exercise in the next lesson.