- Solution

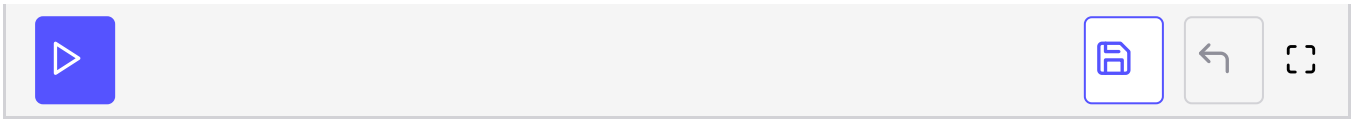
In this lesson, we will discuss the solution to the exercise in of the previous lesson.

WE'LL COVER THE FOLLOWING

Solution

```
//singletonMultithreading.cpp

#include <iostream>
#include <mutex>

class MySingleton{

  private:
    static std::once_flag initInstanceFlag;
    static MySingleton* instance;
    MySingleton()= default;
    ~MySingleton()= default;

  public:
    MySingleton(const MySingleton&)= delete;
    MySingleton& operator=(const MySingleton&)= delete;

    static MySingleton* getInstance(){
      std::call_once(initInstanceFlag, MySingleton::initSingleton);
      return instance;
    }

    static void initSingleton(){
      instance= new MySingleton();
    }
};

MySingleton* MySingleton::instance= nullptr;
std::once_flag MySingleton::initInstanceFlag;


int main(){

  std::cout << std::endl;
```

```
  std::cout << "MySingleton::getInstance(): "<< MySingleton::getInstance() << std::endl;
  std::cout << "MySingleton::getInstance(): "<< MySingleton::getInstance() << std::endl;

  std::cout << std::endl;

}
```



Explanation

- Let's first review the static `std::once_flag` which is declared in line 9 and initialized in line 29.
- The static method `getInstance` (lines 18 - 21) uses the flag `initInstanceFlag` to ensure that the static method `initSingleton` (line 23 - 25) is executed exactly once. The singleton is created in the body of the method.

> **`default` and `delete`**
>
> You can request special methods from the compiler by using the keyword `default`. These methods are special because they are created by the compiler. If we annotate a method with `delete`, the compiler-generated methods will not be available and, therefore, cannot be called. If you try to use these methods, you will get a compile-time error.

> Here are the more details for the keywords default and delete.

- The `MySingleton::getIstance()` method displays the address of the singleton.

For further information:

- std::call_once
- std::once_flag
- Double-Checked Locking Pattern

In the next lesson, we will look at the thread-local data.