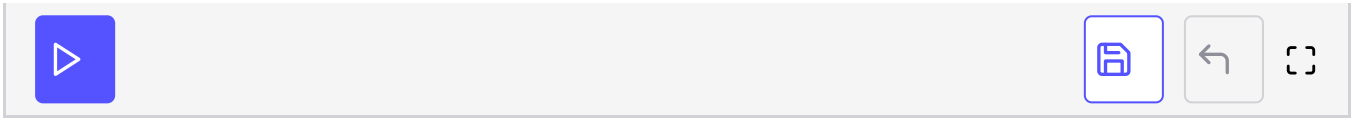
Lifetime Issues of Variables

This lesson explains challenges related to lifetime issues of variables in C++.

Creating a C++ example with lifetime related issues is quite easy. Let the created thread `t` run in the background (i.e. it was detached with a call to `t.detach()`) and let it be only half completed; the creator thread will not wait until its child is done. In this case, you have to be extremely careful not to use anything in the child thread that belongs to the creator thread.

```
// lifetimeIssues.cpp

#include <iostream>
#include <string>
#include <thread>

int main(){

  std::cout << "Begin:" << std::endl;

  std::string mess{"Child thread"};

  std::thread t([&mess]{ std::cout << mess << std::endl;});
  t.detach();

  std::cout << "End:" << std::endl;

}
```



This is too simple. The thread `t` is using `std::cout` and the variable `mess` - both of which belong to the main thread. The effect is that we don't see the output of the child thread in the second run. Only "Begin:" (line 9) and "End:" (line 16) are printed.

In order to see the output of child thread, creator thread will have to wait for it.

Let's see the solution:

```
// lifetimeIssuesSolution.cpp

#include <iostream>
#include <string>
#include <thread>

int main(){

  std::cout << "Begin:" << std::endl;

  std::string mess{"Child thread"};

  std::thread t([&mess]{ std::cout << mess << std::endl;});
  t.join();

  std::cout << "End:" << std::endl;

}
```