The C++ Thread Class Solutions

Pausing a thread

- Modify the "Hello thread" program so that the thread pauses for two seconds before printing out the message
  - The source code for the solutions is in a separate downloadable resource

std::thread ID

- Rewrite the "hello thread" program so it prints out the ID of the worker thread in the hello() function
- Modify the main() function to print out its own ID
- Modify the main() function to print out the ID of the hello thread
  - Before calling join()
  - After calling join()
- Explain your results

std::thread ID

- Explain your results
  - On my system, the output was

```
Main thread has ID 1
Hello thread has ID 2
Hello from thread with ID 2
Hello thread has ID thread::id of a non-executing thread
```

- The main thread and the hello thread have different ID's as required
- When t.join() returns, there is no longer a system thread associated with t
- In that case, get_id() returns a default value

std::thread objects and functions

- Rewrite the "Hello thread" program by adding a function that takes a std::thread object as argument and prints out the object's thread ID
- Pass the std::thread object created in main to this function
- Where, if anywhere, should join() be called on this object?
  - join() should be called to prevent the program terminating before the thread has completed
  - When main passes the thread object to the function, it relinquishes ownership of it
  - join() should be called by the final owner of the thread object, i.e the function it was moved into

std::thread objects and functions (contd)

- Rewrite the "Hello thread" program by adding a function that returns an std::thread object with hello() as its entry point
- Call this function in main
- Print out the ID of the returned std::thread object
- Where, if anywhere, should join() be called on this object?
  - join() should be called to prevent the program terminating before the thread has completed
  - When the function returns the thread object, it relinquishes ownership of it
  - join() should be called by the final owner of the thread object, i.e the function it was returned to

std::thread and exceptions

- Rewrite the "Hello Thread" example so that the thread function throws an unhandled exception
  - What happens?
- Add a handler for the exception to the main() function
  - What happens?
- Move the handler for the exception into the thread function
  - What happens?
- Explain your observations

std::thread and exceptions

- Rewrite the "Hello Thread" example so that the thread function throws an unhandled exception
  - The call stack for the thread is unwound
  - No suitable handler is found, so std::terminate() is called
- Add a handler for the exception to the main() function
  - The call stack for the thread is unwound
  - No suitable handler is found, so std::terminate() is called
- Move the handler for the exception into the thread function
  - The call stack for the thread is unwound
  - A suitable handler is found
  - The exception is caught and the thread (and the program) continue to execute