C++ Thread Class Solutions

Passing a std::thread Object

- Rewrite the "Hello thread" program by adding a function
  - The function takes a std::thread object as argument
  - It displays the object's thread ID
- Pass the std::thread object created in main() to this function

Passing a std::thread Object

- Where, if anywhere, should join() be called?
  - join() should be called
  - The system thread must complete before the object's destructor is called
  - When main() passes the thread object to the function, it releases ownership of the system thread
  - The function argument acquires ownership of the system thread
  - The function is now responsible for calling join() on its argument
  - join() should not be called on the object in main()
  - That object is no longer associated with any system thread

Returning a std::thread Object

- 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
- Display the ID of the returned std::thread object

Returning a std::thread Object

- Where, if anywhere, should join() be called?
  - join() should be called
  - The system thread must complete before the object's destructor is called
  - When main() receives the thread object from the function, it acquires ownership of the system thread
  - main() is now responsible for calling join() on its object
- If the function calls join()
  - The function will stop and wait until the system thread has completed
  - main() will receive an empty object, which is not associated with any system thread

Threads 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

Threads and Exceptions

- Rewrite the "Hello Thread" example so that the thread function throws an unhandled exception
  - The thread's execution stack is unwound
  - No suitable handler is found
  - The entire program is terminated (by default)
- Add a handler for the exception to the main() function
  - The thread's execution stack is unwound
  - No suitable handler is found
  - The entire program is terminated (by default)
- Move the handler for the exception into the thread's task function
  - The thread's execution stack is unwound
  - A suitable handler is found
  - The exception is caught
  - The program continues running