Monitor Class Solutions

Monitor Class

- Briefly describe what is meant by a "monitor" class
  - A monitor class is internally synchronized
  - It performs all synchronization operations necessary to prevent a data race

Naive Solution Drawbacks

- A monitor class can be implemented as follows:
  - A mutex object as private data member
  - Each member function locks and unlocks the mutex when necessary
- Write a simple monitor class using this approach
- Write a simple program to exercise your class

Naive Solution Drawbacks

- Give some drawbacks to this approach
  - If member functions call other member functions, a thread may hold multiple locks. This risks deadlock
  - Transactions may involve multiple member function calls. These require many locking and unlocking operations, which are time consuming. They also create opportunities for race conditions and data races
  - When using it with an existing class, the code needs to be modified

Wrapper Class

- Another option is to write a wrapper class around the object
  - The mutex is a member of the wrapper class
  - The wrapper class has the same public interface as the object
  - Its member functions lock the mutex before calling the object's member function
- Write a simple monitor class using this approach
- Write a simple program to exercise your class

Wrapper Class Pros and Cons

- Give some advantages and disadvantages of this approach
  - It works with any type, including classes that were not designed for threaded code
  - No modifications are needed to the wrapped class
  - Possibility of deadlock due to multiple locking
  - It does not help callers who want to perform transactions. We still have the problems of unnecessary locking operations, which are time consuming and create opportunities for race conditions and data races