Thread-Safe Interface

The thread-safe interface extends the critical scope to an object.

- Antipattern: Each member functions uses a lock internally.
  - The performance of the system is limited.
  - Deadlocks can occur when member functions call each other.

Thread-Safe Interface

A deadlock due to interleaved member functions calls.

```
struct Critical{
    void memberFunction1(){
        lock(mut);
        memberFunction2();
        . . .
    }
    void memberFunction2(){
        lock(mut);
        . . .
    }
    mutex mut;
}
```

```
int main(){
    Critical crit;
    crit.memberFunction1();
}
```

Thread-Safe Interface

- Solution
  - All interface member functions (`public`) should use a lock
  - All implementation member functions (`protected` and `private`) must not use locks
  - The interface member functions only call implementation member functions

threadSafeInterface.cpp