Strategized Locking

Strategized Locking

- makes it possible to use various locking strategies as interchangeable components.
- applys the Strategy Pattern to locking.

- Idea:
  - You want to use your library in different domains
  - Depending on the domain, exclusive locking, shared locking, or no locking should be used
  - Inject your locking strategy at run time or compile time

Strategized Locking

Advantages:

- Runtime polymorphism
  - Allows changing the locking strategy during runtime
  - Is easier to understand for developers with an object-oriented background
- Compile-time polymorphism
  - Has no costs at runtime
  - Has a flat hierarchy

Disadvantages:

- Runtime polymorphism
  - Needs a pointer indirection
  - Can have a deep object hierarchy
- Compile-time polymorphism
  - Can generate error messages that are difficult to understand

strategizedLockingRuntime.cpp
strategizedLockingCompiletime.cpp