Summary of Part I

We've covered a lot of material so far! The following "concurrency cheat sheet" summarizes the main concepts and rules presented in Part I.

- *It's the mutable state, stupid.*[1]

  All concurrency issues boil down to coordinating access to mutable state. The less mutable state, the easier it is to ensure thread safety.

- *Make fields final unless they need to be mutable.*

- *Immutable objects are automatically thread-safe.*

  Immutable objects simplify concurrent programming tremendously. They are simpler and safer, and can be shared freely without locking or defensive copying.

- *Encapsulation makes it practical to manage the complexity.*

  You could write a thread-safe program with all data stored in global variables, but why would you want to? Encapsulating data within objects makes it easier to preserve their invariants; encapsulating synchronization within objects makes it easier to comply with their synchronization policy.

- *Guard each mutable variable with a lock.*

- *Guard all variables in an invariant with the same lock.*

- *Hold locks for the duration of compound actions.*

- *A program that accesses a mutable variable from multiple threads without synchronization is a broken program.*

- *Don't rely on clever reasoning about why you don't need to synchronize.*

- *Include thread safety in the design process—or explicitly document that your class is not thread-safe.*

- *Document your synchronization policy.*

1. During the 1992 U.S. presidential election, electoral strategist James Carville hung a sign in Bill Clinton's campaign headquarters reading "The economy, stupid", to keep the campaign on message.