1. Intro to concurrency and the mutual exclusion problem

- Define: Multiple overlapping streams of program statements.
  - Sequential vs Parallel vs Concurrent
- Motivation
  - Exploitation
  - Inherent
  - Hidden (Virtual)
- Data race:
  - Shared data
  - At least one write
- Critical Section:
  - Mutual exclusion
- Race condition:
  - Interleaving

2. Synchronization

- Motivation → Shared memory:
  - Mutual exclusion.
  - Visibility.
- Locks:
  - Mutual exclusion, visibility, prevents reordering.
  - Intrinsic locks.
  - Reentrant lock.
- Monitors:
  - Internal state, methods (mutex), condition variables.
  - Implemented as classes in OO.
  - Fairness.
- Semaphores:
  - Permits.
  - Limit concurrent access to shared ressource.

3. Visibility

- Data dependencies – Example
- Reordering - Example
- Happens before
- Volatile
  - May not be reordered.
  - Not stored in registers
  - Doesn't ensure mutex.
- Code printout and code from exercises.

4. Thread-safe classes

- Absence of data races on fields and methods.
  - Class state
    - Private fields.
    - Avoid fields from parent class.
  - Escaping
    - Never return reference – Copy.
  - (Safe) publication
    - Visibility issues.
    - Final, Volatile, Default values, Static, AtomicReference.
  - Immutability
    - No modification to fields, Safe publication, State doesn't escape.
  - Mutual exclusion
    - Shared mutable state.
- Thread-safe program: race condition free.

5. Testing

- Undesired interleavings: counterexamples.
- Properties:
  - Safety
  - Liveness
- Strategies:
  - Property
  - Interface
  - Barriers
  - RepeatedTest(X)
- Deadlocks.

6. Performance measurements

- Exploitation.
- Time consuming tasks:
  - Searching, computing prime numbers.
- Better research.
- Normal distribution – Standard deviation

- Pitfalls:
    - JIT.
    - Iterations.
    - Dead code.
    - See Q7 pitfalls for more.

7. Performance and Scalability

- Executor, Future, and Threads
- Amdahl's Law
    - Speed up = 1 / (F + (1-F) / N)
- Pitfalls (Loss):
    - Starvation.
    - Separation.
    - Saturation.
    - Braking.
- Locking large data structures – E.g. lock striping for hashmaps.

8. Lock-free Data Structures

- CAS typically faster, but higher memory usage.
- Lock-free data structures
- AtomicXX
- ABA problem
- Code printout and code from exercises.

9. Linearizability

- Motivation: Arguing about concurrent objects.
- Sequential consistency (show example).
- Linearizability
    - Preserves real time order of execution.
    - Linearization point.
    - Object is linearizable iff. all executions are linearizable.

10. Streams

- Motivation:
    - Exploitation.
    - Embarrasingly parallel tasks.
- Java Streams – Pull based:
    - Sources – parallelizeable – may be infinite.
    - Intermediate – Lambda expressions.
    - Terminal – Laziness.
- Pitfalls:
    - Stateful lambdas.
    - Associativity in parallel reductions.
    - Terminals on infinite streams.
- RxJava – Push based:
    - Oberservable
    - Observer
    - Backpressure

11. Message Passing

- Motivation:
    - Shared state is dangerous – Especially mutable.
    - Sharing through messages.
- Actor model - Actors:
    - Build in message passing.
    - Draw model.
    - What is an actor?
        - Sequential unit of computation (abstraction of thread).
        - Actions: Receive, Send, Create, Change.
        - Mail address, Local state, Mailbox.
    - Messages:
        - No guarantee of order.
        - Send is non-blocking – receive is blocking.
- Topology:
    - Number of actors.
    - Communication between actors.
    - Static vs. Dynamic.
    - Load Balancing