JVM Thread Communication

Thread Comms: Exercises!

Synchronized

Entering a synchronized expression on an object *locks the object:*

```
val someObject = "hello"

someObject.synchronized {
  // code
}
```

lock the object's *monitor*

any other thread trying to run this will block

release the lock

Only AnyRefs can have synchronized blocks.

General principles:

- make no assumptions about who gets the lock first
- keep locking to a minimum
- maintain *thread safety* at ALL times in parallel applications

wait() and *notify()*

wait() –ing on an object's monitor suspends you (the thread) indefinitely

```
// thread 1
val someObject = "hello"
someObject.synchronized {
  // ... code part 1
  someObject.wait()
  // ... code part 2
}
```

lock the object's *monitor*

release the lock and... wait

when allowed to proceed,
lock the monitor again and continue

```
// thread 2
someObject.synchronized {
  // ... code
  someObject.notify()
  // ... more code
}
```

lock the object's *monitor*

signal ONE sleeping thread they may continue

but only after I'm done and unlock the monitor

Which thread?
You don't know!

Use *notifyAll()*
to awaken ALL threads

Waiting and notifying only work in *synchronized* expressions.

Scala rocks