JVM Thread Communication

Synchronized

Temporarily lock an object from multi-threaded access

```
val someObject = "hello"

someObject.synchronized {

  // code

}
```

lock this object's *monitor*

any other thread trying to call `synchronized` will block

the object's monitor is released

General principles

- make no assumptions about which thread gets the lock first
- keep locking to the minimum necessary
- always maintain thread safety: eliminate race conditions, deadlocks, livelocks

wait() and notify()

Waiting on an object's monitor suspends calling 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 suspend

when notified by another thread,
re-acquire the lock and continue

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

lock the object's monitor

signal one waiting thread to continue

the notified thread will continue
after it acquires the lock

Which thread?
You don't know!

Use *notifyAll()*
to awaken all waiting threads

Scala rocks