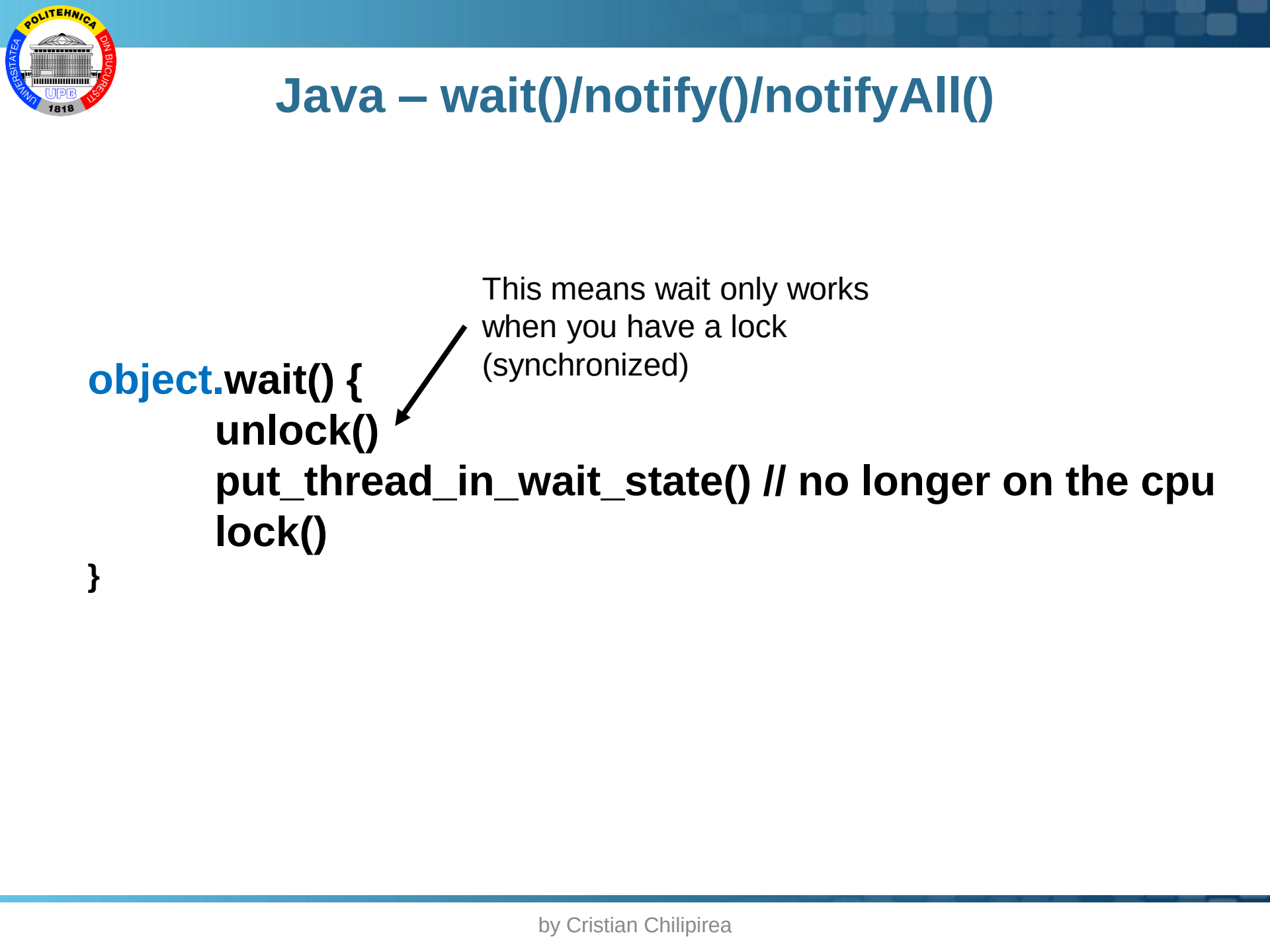
Java – wait()/notify()/notifyAll()

```
this.wait() {
        unlock()
        put_thread_in_wait_state() // no longer on the CPU
        lock()
}
```

Java – wait()/notify()/notifyAll()



Java – wait()/notify()/notifyAll()

```
this.
object.notify() {
        thread = pick_random_thread_waiting_on_object()
        move_from_wait_to_running(thread)
}
```

Java – wait()/notify()/notifyAll()

this.notifyAll()
object.notifyAll()

Puts all threads that wait on object in running state.
Still only one manages to get the lock()

Java – Atomic*

- AtomicInteger
- AtomicLong
- AtomicBoolean
- AtomicFloat
- AtomicReference

- Problem: 32bit/64bit systems (+ other architectures)
  - Copying an element of 64bit in a 32 bit system takes two operations
  - Someone can read only part of a variable

- Atomic* makes sure all operations are atomic (no in between operation reads)
- Offers getAndSet/getAndIncrement type atomic operations