LOYOLA
UNIVERSITY
CHICAGO

COMP 413: Intermediate Object-Oriented Programming

Dr. Robert Yacobellis
Advanced Lecturer
Department of Computer Science

Week 14 Topics

- **Test 3 – 60 minutes, open book/notes**
- Other topics – see online schedule for links:
  - More on concurrency
  - Prime Number Checker (asynchronous execution)
- Possibly time to work on Project 4
- Reminders:
  - The COMP 413 IDEA survey opened November 28 and closes on December 11 at 11:59pm
  - Quiz 3 next week (SE Radio #65, 12, 23)
  - Test 4 (final) December 13 – practice Test is on Sakai

Week 14 Topics

- **Other topics – see online schedule for links:**
  - **More on concurrency**
  - Prime Number Checker (asynchronous execution)
- Time to work on Project 4

Concurrency and Interleaving

- Suppose we have two parallel threads of Java code:

Thread 1

```
int counter;
// shared instance variable
…
counter++;
```

Thread 2

```
int counter;
// shared instance variable
…
counter--;
```

Assuming that `counter` initially has value **0**, what are the possible values of `counter` after both threads run to completion in parallel, and why?

- **Answer: -1, 0, or 1!**

Concurrency and Interleaving

- Conceptually, the threads perform these operations:

Thread 1

```
…
// counter++;
fetch counter (f1)
increment (i1)
store counter (s1)
```

Thread 2

```
…
counter--;
fetch counter (f2)
decrement (d2)
store counter (s2)
```

Assuming those operations are atomic, the threads can be interrupted before or after any of them.

- **Value -1: *f1 f2 i1 s1 d2 s2 (and many other sequences)***
- **Value 0: *f1 i1 s1 f2 d2 s2***
- **Value 1: *f1 f2 d2 s2 i1 s1***

Concurrency and Interleaving

- We can prevent this by locking the "critical region":

Thread 1

```
int counter;
// shared instance variable
…
synchronized(this) {
  counter++;
}
```

Thread 2

```
int counter;
// shared instance variable
…
synchronized(this) {
   counter--;
}
```

We can synchronize on any object, not just the current object (`this`) – every object in Java has an associated lock property (and so do Classes!).
We can also use a `Lock` object (Java `Lock` class).

Week 14 Topics

- **Other topics – see online schedule for links:**
  - **More on concurrency**
    - **Physical (hw) vs. logical (sw) concurrency**
    - **CPU-bound vs. I/O-bound activities**
    - **Run to completion vs. coordination**
    - **Conflicting design forces: safety, liveness, performance (throughput, latency, jitter)**
  - Prime Number Checker (asynchronous execution)
- Time to work on Project 5/6

Week 14 Topics

- **Other topics – see online schedule for links:**
  - More on concurrency
  - **Prime Number Checker (asynchronous execution)**
    - **Cloud-based execution (AsyncHttp…)**
    - **Direct execution (AsyncTask, single thread)**
    - **Background execution (AsyncTask, thread pool)**
- Time to work on Project 4

Week 14 Topics

- Other topics – see online schedule for links:
  - More on concurrency
  - Prime Number Checker (asynchronous execution)
- **Time to work on Project 4**