cs2220: Engineering Software

Class 15: Threads and Concurrency

Fall 2010
University of Virginia
Robbie Hott

PS5

- Team requests due tonight by midnight
- Teams of 2-3

Remember:

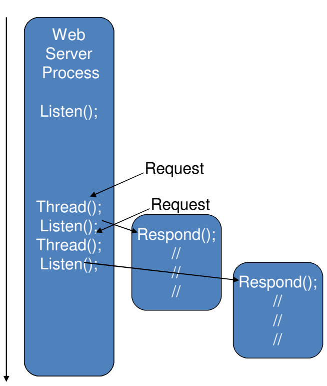
"A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language."

[Sun95]

Concurrent Programming

- Our computer can only do one instruction at a time, why would we want to program pretending it can do many things at once?
- **Concurrency**: having several computations interleaved or executing simultaneously, potentially interacting with each other

Threading Concept



- Multiple Threads of execution at once
- One set of shared data

Concurrent Programming

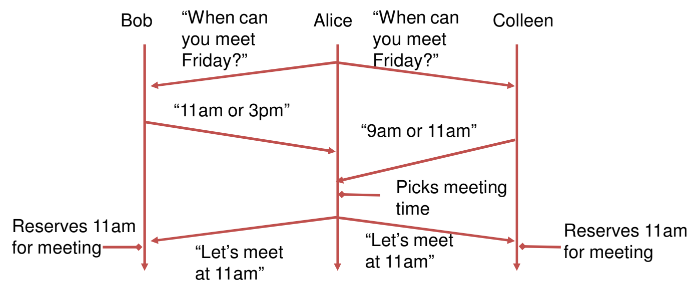
- Why?
- Some problems are clearer to program concurrently
  - Modularity: Don't have to explicitly interleave code for different abstractions (especially: user interfaces)
  - Modeling: Closer map to real world problems: things in the real world aren't sequential

Simple Example: Counter (in Java)

- One Counter with two operations, increment and decrement.
- Two Threads, one calls increment, the other calls decrement.
- After each call, they sleep.
- What do you think will happen?

Example: Scheduling Meetings

Alice wants to schedule a meeting with Bob and Colleen

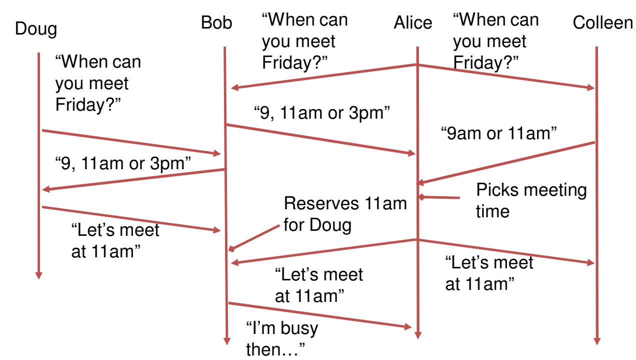


Partial Ordering of Events

- Sequential programs give use a *total ordering* of events: everything happens in a determined order
- Concurrency gives us a *partial ordering* of events: we know some things happen before other things, but not total order

Alice asks to schedule meeting before Bob replies
Alice asks to schedule meeting before Colleen replies
Bob and Colleen both reply before Alice picks meeting time
Alice picks meeting time before Bob reserves time on calendar

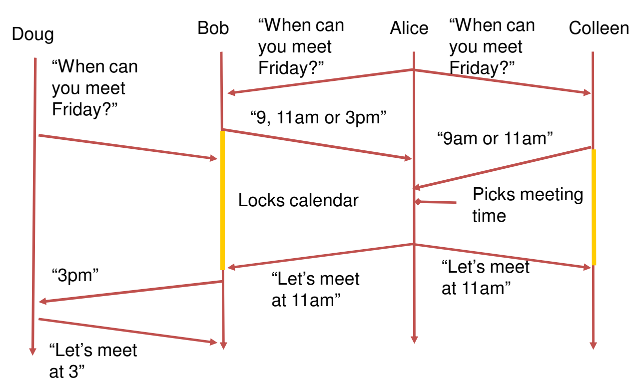
Race Condition



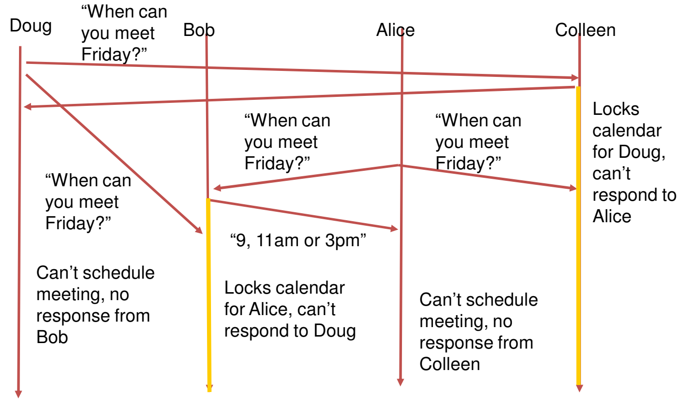
Preventing Race Conditions

- Use locks to impose ordering constraints
- After responding to Alice, Bob reserves all the times in his response until he hears back (and then frees the other times)

Locking



Deadlocks



Deadlocks

- **Deadlock:** when computation has stalled because execution units are blocked and waiting on a circular dependency chain. For example, when 2 or more threads wait for the other's response to finish. Therefore, neither does.
  - Other examples?
    - "When two trains approach each other at a crossing, both shall come to a full stop and neither shall start up again until the other has gone."
      —statute passed by the Kansas Legislature (wikipedia)

Concurrency in Java

```
public class Thread implements Runnable {
  // OVERVIEW: A thread is a thread of execution in a program.
  //     The Java Virtual Machine allows an application to have
  //     multiple threads of execution running concurrently.

  public Thread (Runnable target)
      // Creates a new Thread object that will run the target.

  public void start ()
      // Starts a new thread of execution. Calls the target's run().

  … many other methods
}
```


Concurrency in Java

public interface **Runnable {**
  public void **run**()
    When an object implementing interface Runnable is used to create a thread, starting the thread causes the object's run method to be called in that separately executing thread. **The general contract of the method run is that it may take any action whatsoever.**
  }

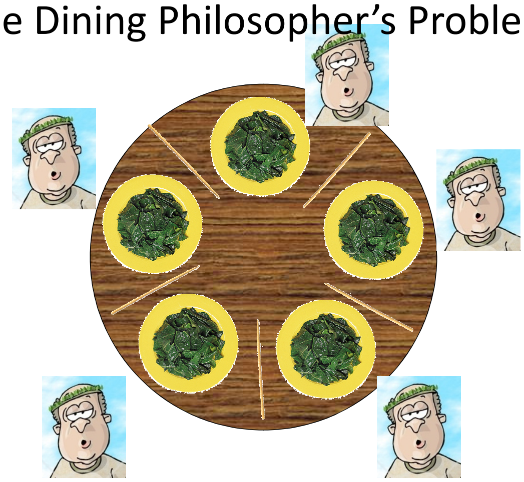
Simple Java Example: Counter

- One Counter with two operations, increment and decrement.
- Two Threads, one calls increment, the other calls decrement.
- After each call, they sleep.
- What do you think will happen?

Why are threads hard?

- Too few ordering constraints: race conditions
- Too many ordering constraints: deadlocks
- Hard/impossible to reason modularly
  - If an object is accessible to multiple threads, need to think about what any of those threads could do at any time!
- Testing is even more impossible than it is for sequential code
  - Even if you test all the inputs, don't know it will work if threads run in different order

The Dining Philosopher's Problem



The Dining Philosopher's Problem

- What are the issues to avoid?
  - Deadlock
  - Starvation

The Dining Philosopher's Problem

- How does it look in Java?