Multithreading

in honour* of the

Royal Wedding

*spelled correctly
Extensions to project 5

We'll implement a couple of fun things (insofar as we have time):

Java-style joins.

Starter thread waiting on spawned thread completion.

Thread communication

Via the producer/consumer metaphor

A rather extreme case of multithreading.
In Java:

```java
private void threadTest(int numOfThreads) {
    Thread[] threads = new Thread[numOfThreads];
    for (int i = 0; i < threads.length; i++) {
        threads[i] = new foo.ThreadTest.MyThread();
        threads[i].start();
    }

    for (int i = 0; i < threads.length; i++) {
        try {
            threads[i].join();
        } catch (InterruptedException ignore) {} 
    }
}

// after http://javahowto.blogspot.com/2007/05/when-to-join-threads.html

Matrimony m = new Matrimony("Wills", "Kate");
m.join();
```

In p5 world: we'll define a command **Join of expr** and augment **step** to wait on a label when **Join n** is demanded.

Have to keep track of ids of spawned threads...

Author: Asad Sayeed
Producer/consumer

Existing p5 wait/notify semantics support:

Producer thread: fills slots with integers
Consumer thread: does something with integers

So what we'll do is:

Expand the expression syntax to generate random numbers
Expand the expression syntax to do print side-effects.
Write a program in our language that generates and prints random numbers across two threads

Bonus: build thread communication into the cmds.

Author: Asad Sayeed