Concurrency in Java, Taming Threads

July 10, 2017

Overview For the Class

- Review of threads (good, bad and the ugly)
- Ways to deal with threads in Java
  - synchronize
  - parallelStream()
  - async IO (Probably Monday…)

Threading

Threading

"~~Beer~~ Threading, the cause of, and solution to, all of our problems."

- Homer Simpson

Treading

- Processors aren't getting (much…) faster
- Speed through parallelism
- Shared memory

How does this differ from subprocesses?

Danger of Threads

- Same code is being executed at the same time
- Order is unpredictable
- Unpredictability leads to integrity problems

threading-trouble/{Main, Account}.java –>

Synchronize

- Tells java that only one thread can execute this method at a time
- Like a lock / semaphore in C
- per method, per object

synchronize-example/
{Main, UnSyncExample}.java –>

threading-trouble/{Main, Account}.java –>

Downsides of synchronize

- Its very easy to get wrong
- It undoes the point of threads!

Concluding Threads

- They're useful for speeding up our programs
- They're terrifying
- They're a course in an of themselves
- Java provides some abstractions that makes them less terrible

Streams

Why Streams?

Why Streams?

- Encourages concurrent constructions
- Convenient Syntax
- Easy to parallelize

streams-example/Main.java –>

url-titles/* –>

Final Projects

- Example of finding an issue on GitHub to work on
- Come up with a list of issues (or an application proposal)
- I'll meet with groups individually for the rest of class