

Introduction

- The field of concurrency was started by Dijkstra
- Concurrency: study of systems of interacting computer programs which share resources and run concurrently
 - “at the same time” not exactly at the same time, the OS scheduler switches apps
 - “interacting” exchange information and agree on certain things
 - * car wash, the machine and car wash must interact to let it know where it is and when to do certain things
 - * message passing
 - send/receive
 - * shared memory
 - read/assign to variable or memory location
- Problems with the granny example of one set of needles:
 - DEADLOCK: each granny takes a needle and waits indefinitely until the other one has freed the one she has
 - LIVELOCK: Each granny takes a needle, sees the other granny has the other needle and returns it, repeats indefinitely
 - * Infinite Loop, instructions being executed, system isn’t making any progress (neither granny gets two needles)
 - STARVATION: One of the grannies always takes the needle before the other one
 - There are other concurrency problems that can arise, but these are the most common
 - * Synchronization include other problems (ferry problem)
- `thread.start() <- 1`
`thread.start() <- 2`
 - We don’t know which thread will run first, `thread.start` just spawns thread, scheduler will decide which runs first
- **SEE *l2ex1.groovy***
 - Any combination is possible so long as A comes before B and C comes before D because of context switching between threads and scheduling
 - m are num instructions in first thread
n are num instructions in second thread
of possible interleavings of instructions:

$$\frac{(m+n)!}{m!n!} = \frac{m+n}{n}$$

- Synchronization: making sure it prints ABCD (use semaphores)
- Transition Systems
 - (s, ->, I)
 - * S is a set of states
 - * -> C_ SxS is a transition relation
 - * I C_ S set of initial states
 - *See transition ex1, ex2, ex3, and ex4 on ipad l2*
 - *for more code see l2ex3*
- Can now do exercise booklet
- Quiz Friday