Concurrency in Java

July 10, 2017

Reading Quiz

What is Concurrency?

A. Execute multiple programs (or parts of programs) at the same time.

B. Making a single program run faster through pipelining.

C. Reducing memory use by compressing data automatically.

D. Writing a single program in multiple programming languages.

Which Java code **is not** related to concurrency?

A. new **java.lang.Thread**(() -> {// fake out};)

B. public **synchronized** void yeppers() {};

C. catch (**InterruptedException** error) { }

D. public **final** class PikaPika {}

What is a risk associated with Java's concurrency model?

A. Two threads may try to access the same memory at the same time, leading to an Exception being thrown.

B. Compatibility problems, due to changes in the Java language over time.

C. Two threads may access the same variable in an unpredictable way, leading to correctness issues.

D. Java's concurrency model is very verbose, requiring a lot of typing, making it likely that code will include errors.

What does the "synchronize" modifier for?

A. To try and speed up concurrent programs by batching processor instructions.

B. To try and make Java less fun by requiring more typing.

C. To try and make concurrent programs more memory efficient by sharing resources between threads.

D. To try and prevent integrity errors related to concurrency.

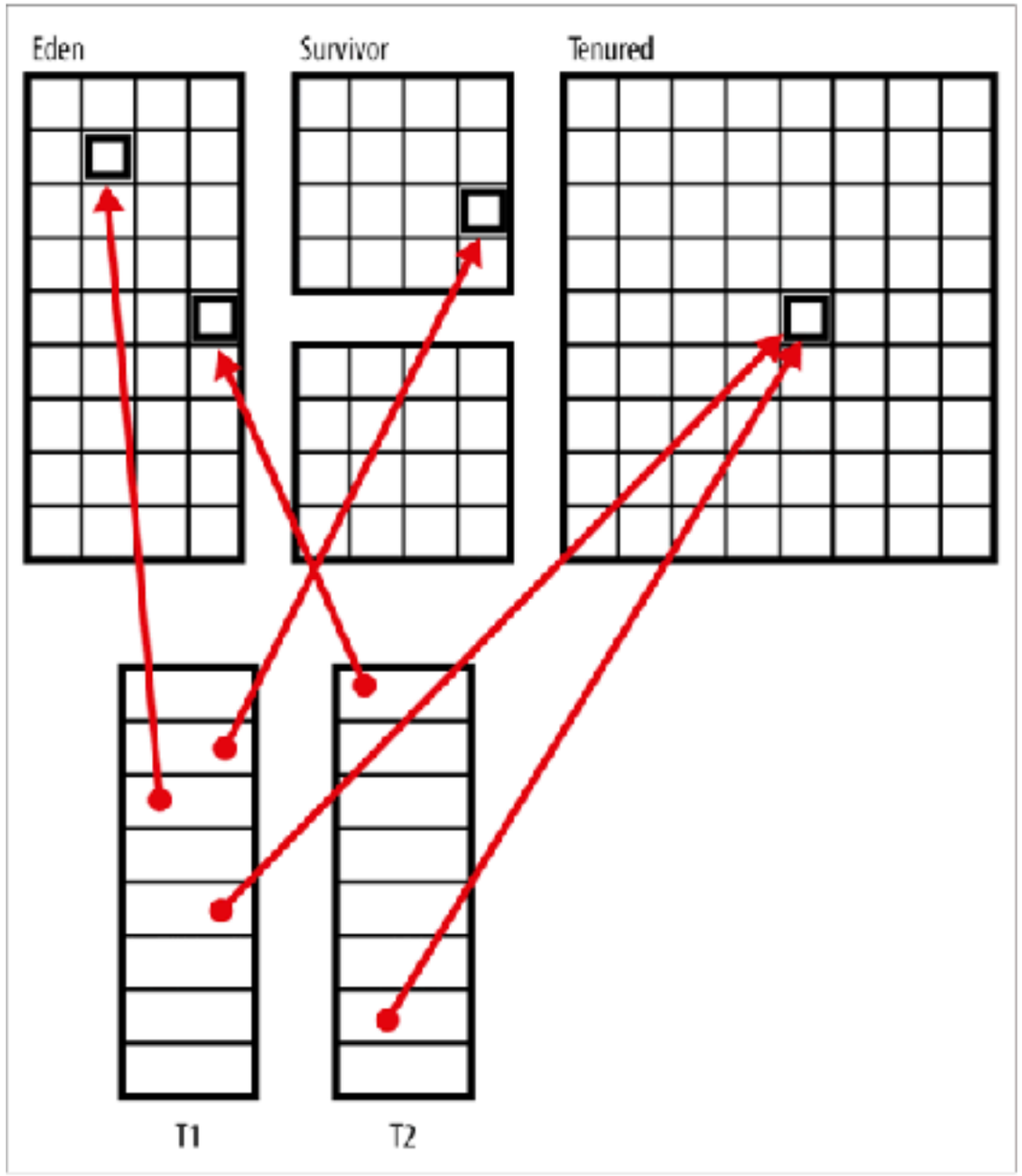
What in the world is being demonstrated in this image from the book?

A. That threads share some kinds of memory, but not others.

B. That Java is great for drawing arrows.

C. How java manages tables in memory.

D. That some threads are of type Tenured, some are of type Survivor, and some are of type Eden.



Homework 5

Homework 5

- Due in a week
- Unit testing / Test driven development
- We'll discuss after the break

Final Project

Final Project

- Due July 31st
- Groups of one or two people
    - Email me groups by class on Wednesday
    - If you want a group, but can't find one, email me
- I'll meet with groups on Wednesday

Final Projects

- Patch to open source project
  - Select your own project, or I can suggest one
  - Must be code
  - Really really good for resume!
- Some non-trivial java application
  - Web application
  - Android application (lots of work!)
  - Fine, but not as good, for a resume

Part One

- Status checks in class with me every other class
- One half to one-page summary of what has been accomplished since previous meeting (specific specific specific!)
- Graded, not just instructive

Part Two

- Ten to fifteen minute class presentation
  - Introduce the project
  - Programming challenges
  - What you learned / would recommend to others

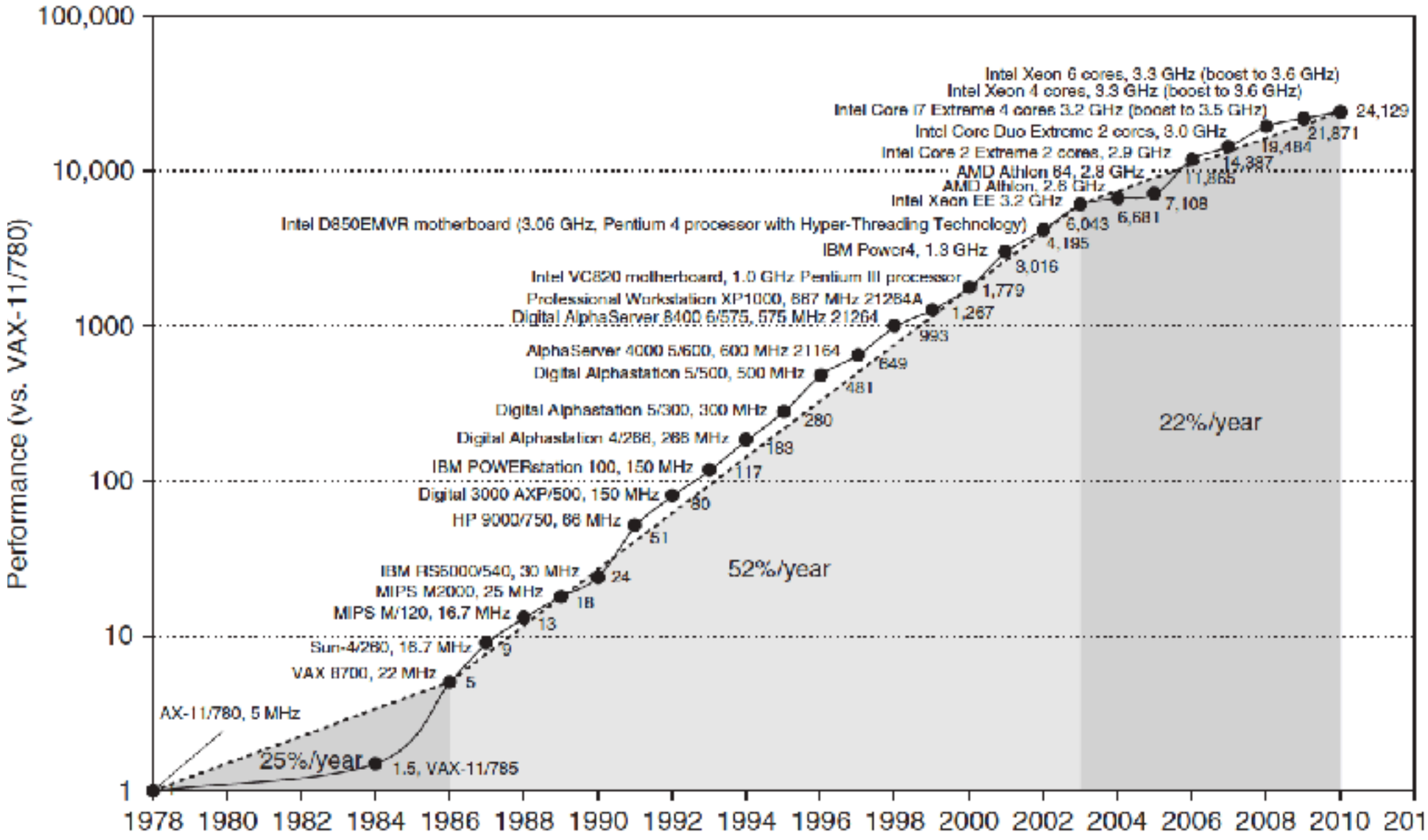
Part Three

- Code / contribution evaluation
    - Code quality
    - Documentation
    - Unit tests
    - Communication with project managers (if applicable)

Concurrency

Problem

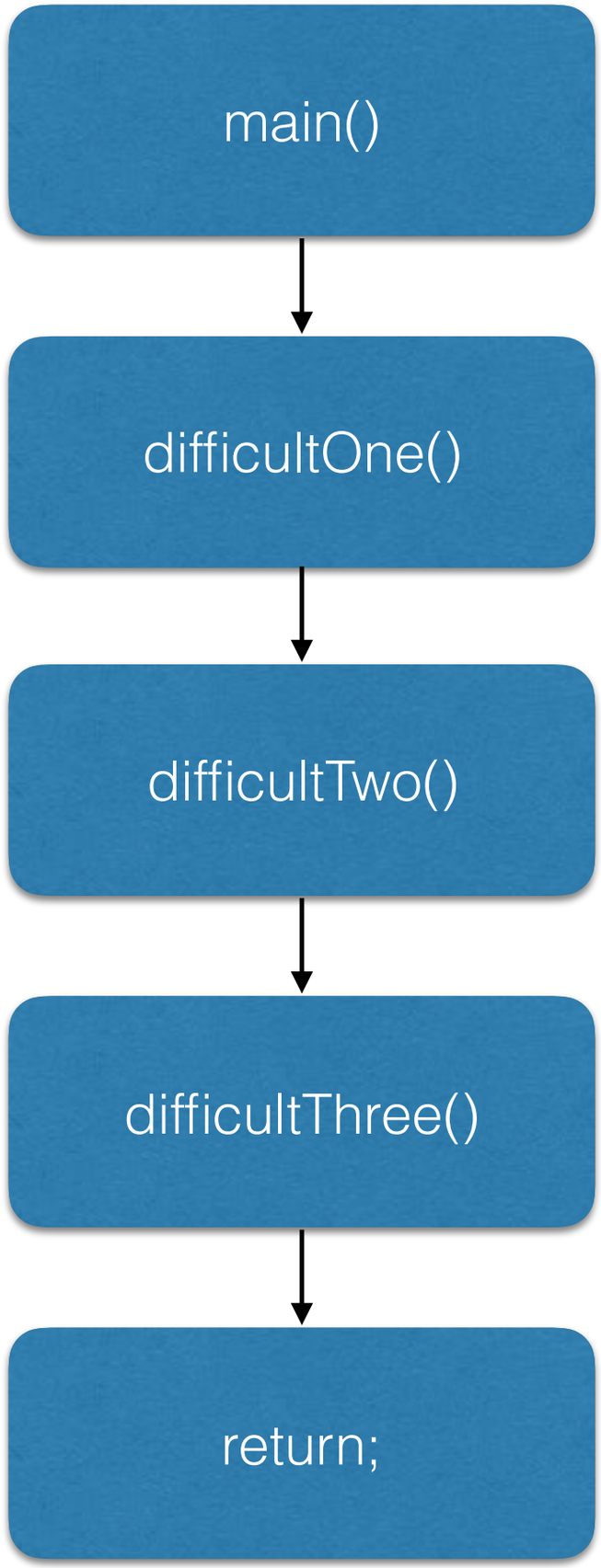
- We want our programs to go fast
- Processors used to get much faster, quickly
- Less and less the case



Processor Speeds over Time

The Speed Problem

- Instead of getting faster, processors now get more complex
- Execute multiple things at the same time
- Performance improvement through waiting less


main()
difficultOne()
difficultTwo()
difficultThree()
return;

problem/Main.java –>

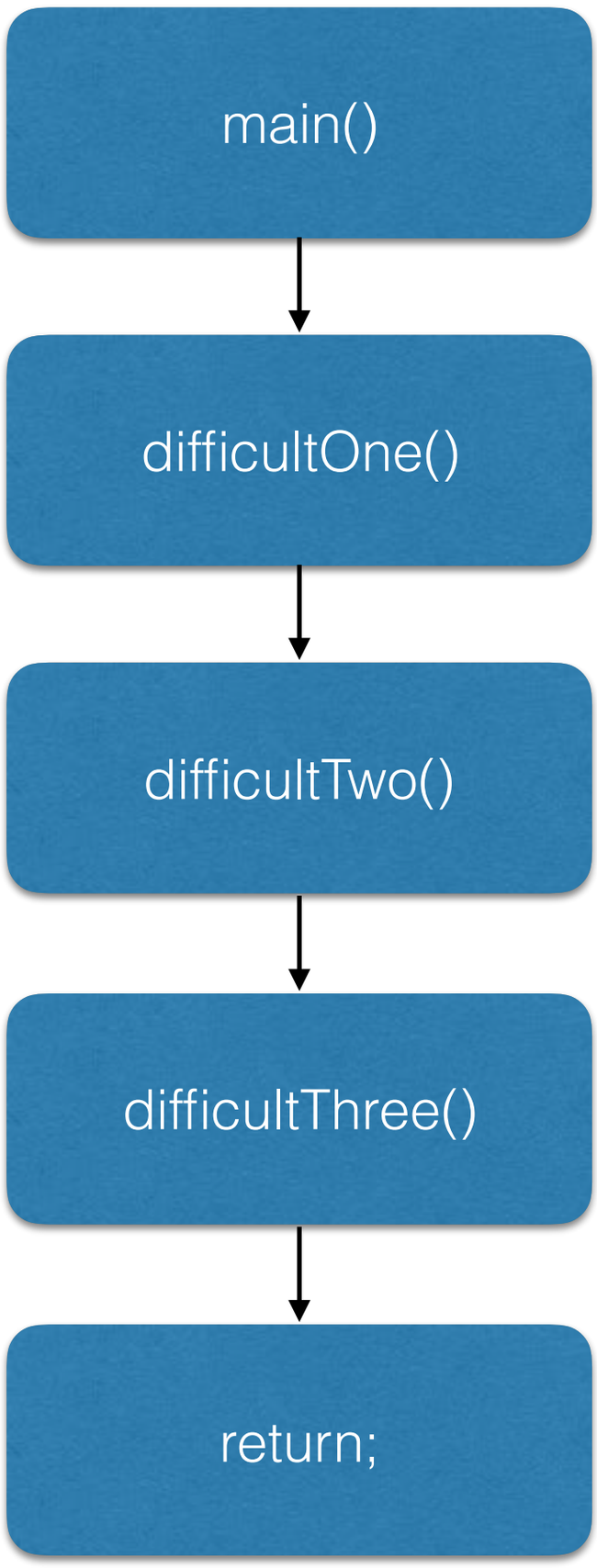
Parallel Execution

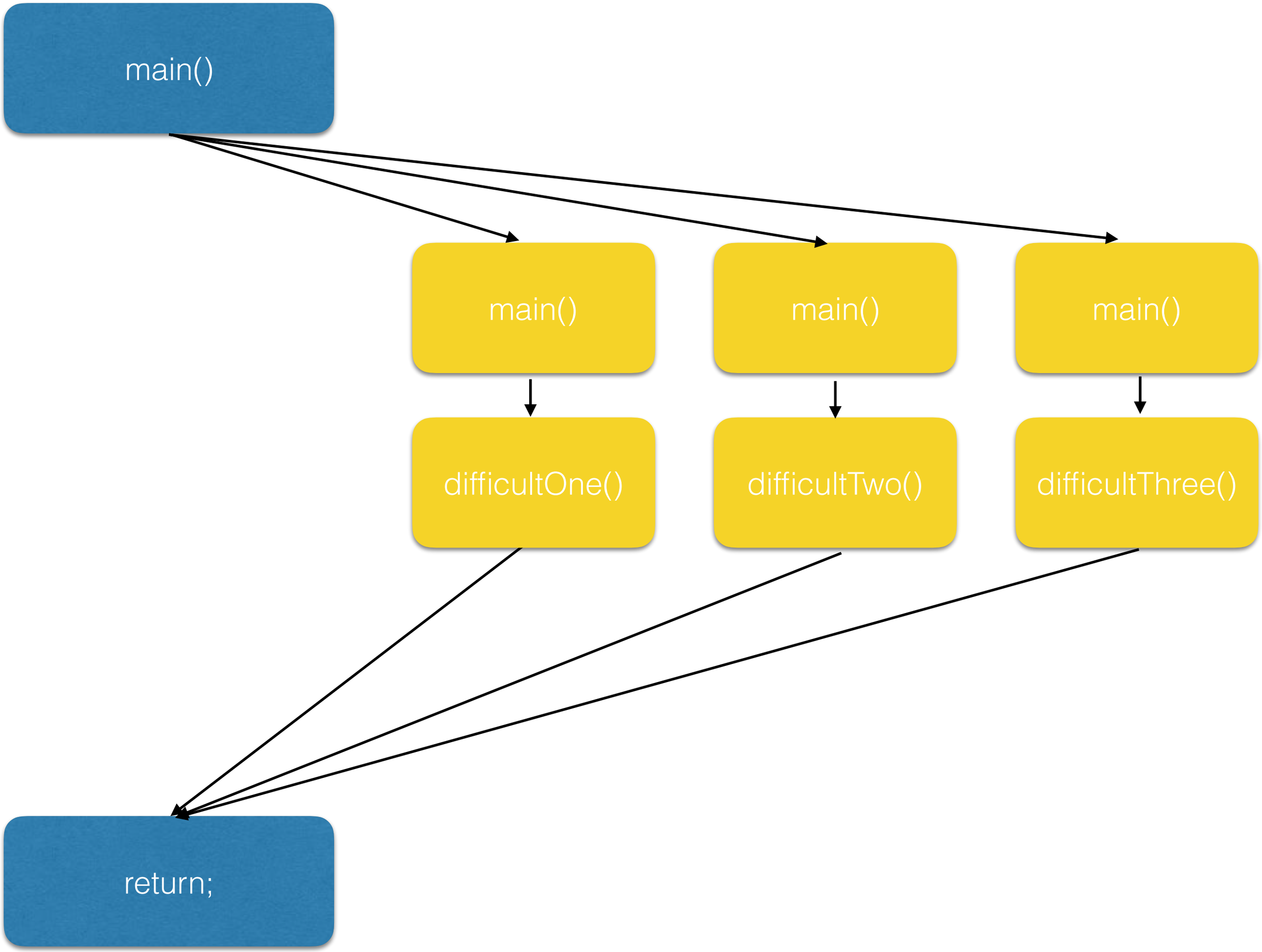
- Subprocesses
- Threads
- Runtime managed threads
  (ie threads, with Java doing the hard stuff)

Subprocesses

Subprocesses

- Write programs that do smaller parts of the task
- Run these child programs (processes) from our program
- Read the results back into our program
- No shared memory


main()
difficultOne()
difficultTwo()
difficultThree()
return;


main()
main()
main()
main()
difficultOne()
difficultTwo()
difficultThree()
return;

Subprocesses in Java

- ProcessBuilder
  Manage connections to subprocesses

  - ProcessBuilder#start: Start running the child process

- Process
  Represents each child process

  - Process#waitFor: Wait until the process finishes

- InputStreamReader
  Read output from child process

subprocess/{Main, Subprocess}.java –>

Subprocess Benefits

- Simple!
- Cross language (subprocess/Main2.java –>)
- Free of all the problems of threading (deadlocks, race conditions, etc.)

Subprocess Costs

- Expensive (kernel has to manage each process)
- Sharing data is difficult (STDIO, STDIN, pipes)
- Slower than alternatives

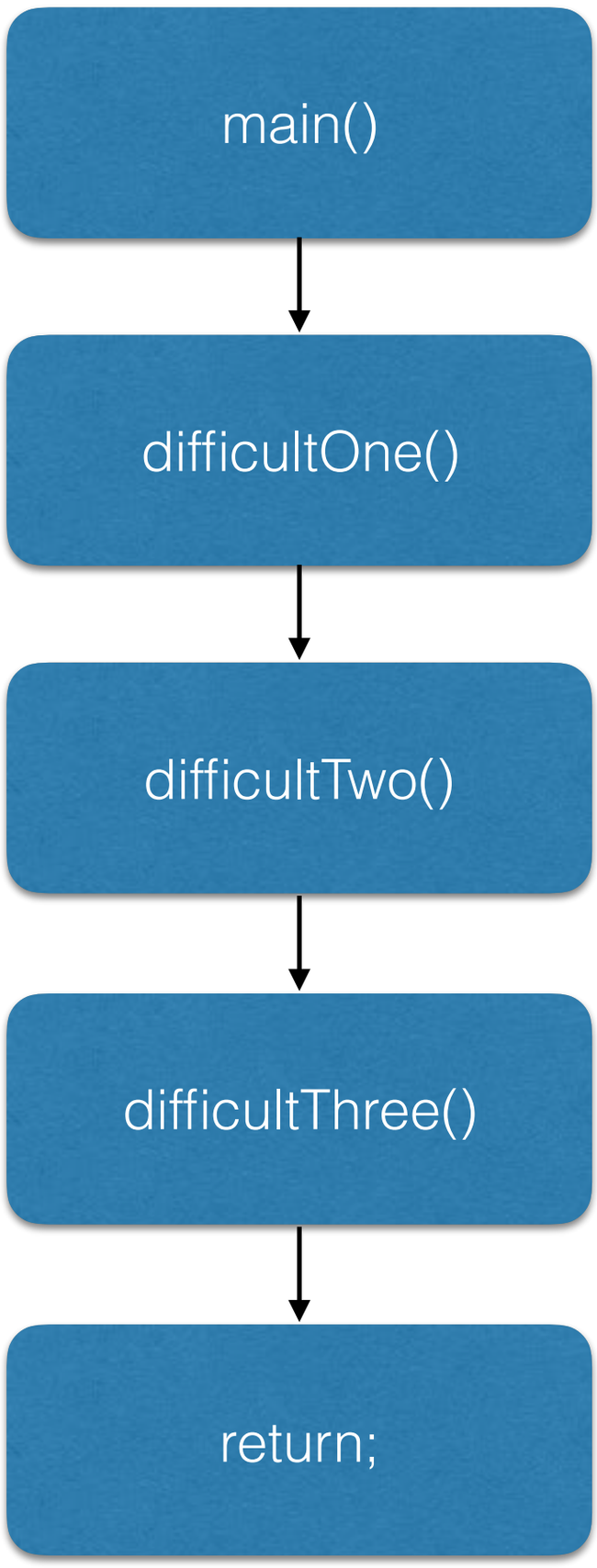
Threads

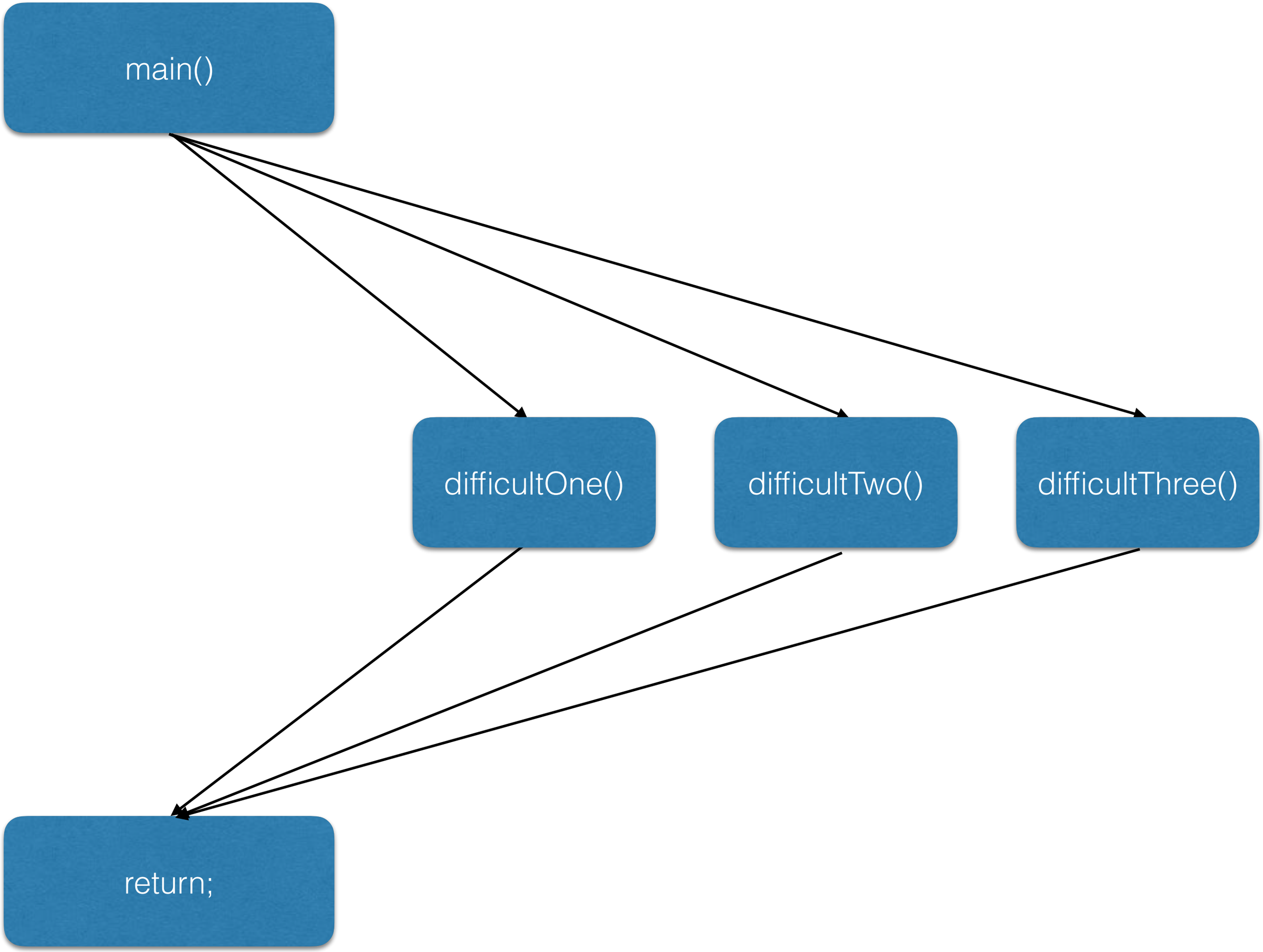
Threads

- "Sub-program" / multiple executions within a program

- Share memory across "sub-programs"

- Instruct java which parts of our program can run at the same time (threads)


main()
difficultOne()
difficultTwo()
difficultThree()
return;


main()
difficultOne()
difficultTwo()
difficultThree()
return;

Threading in Java

- Thread
  Represents a thread of execution

  - Thread::sleep
    Tells the current thread to pause for a while

  - Thread#start
    Starts the constructed thread's execution

  - Thread#join
    Wait for a thread to finish executing

- Runnable
  Interface that represents something that can be run as a thread

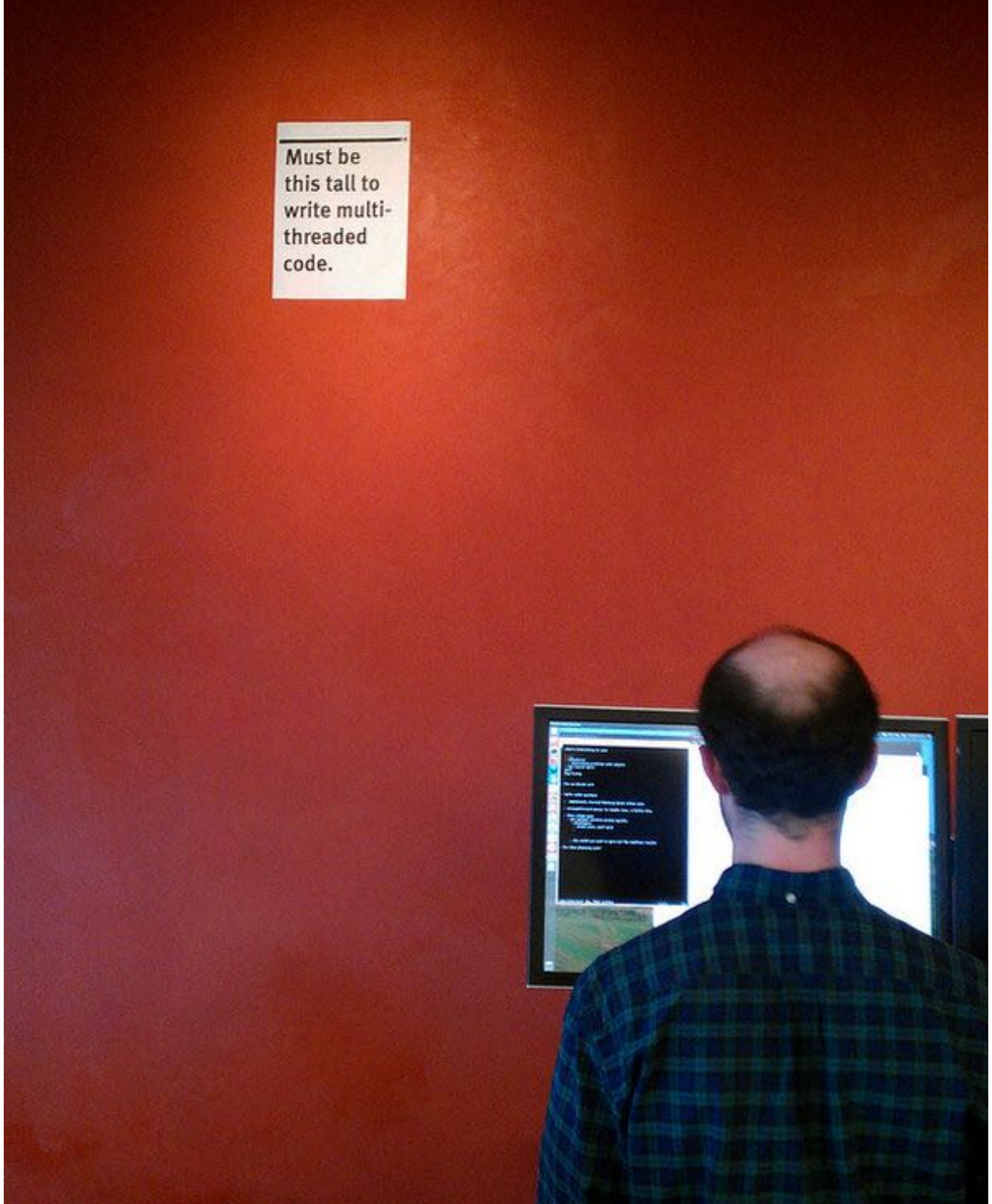
threads/Main.java –>

Threading Benefits

- Fast!
- Share memory (no pushing stuff between child processes, etc.)
- Granular / low memory cost (compared to processes)

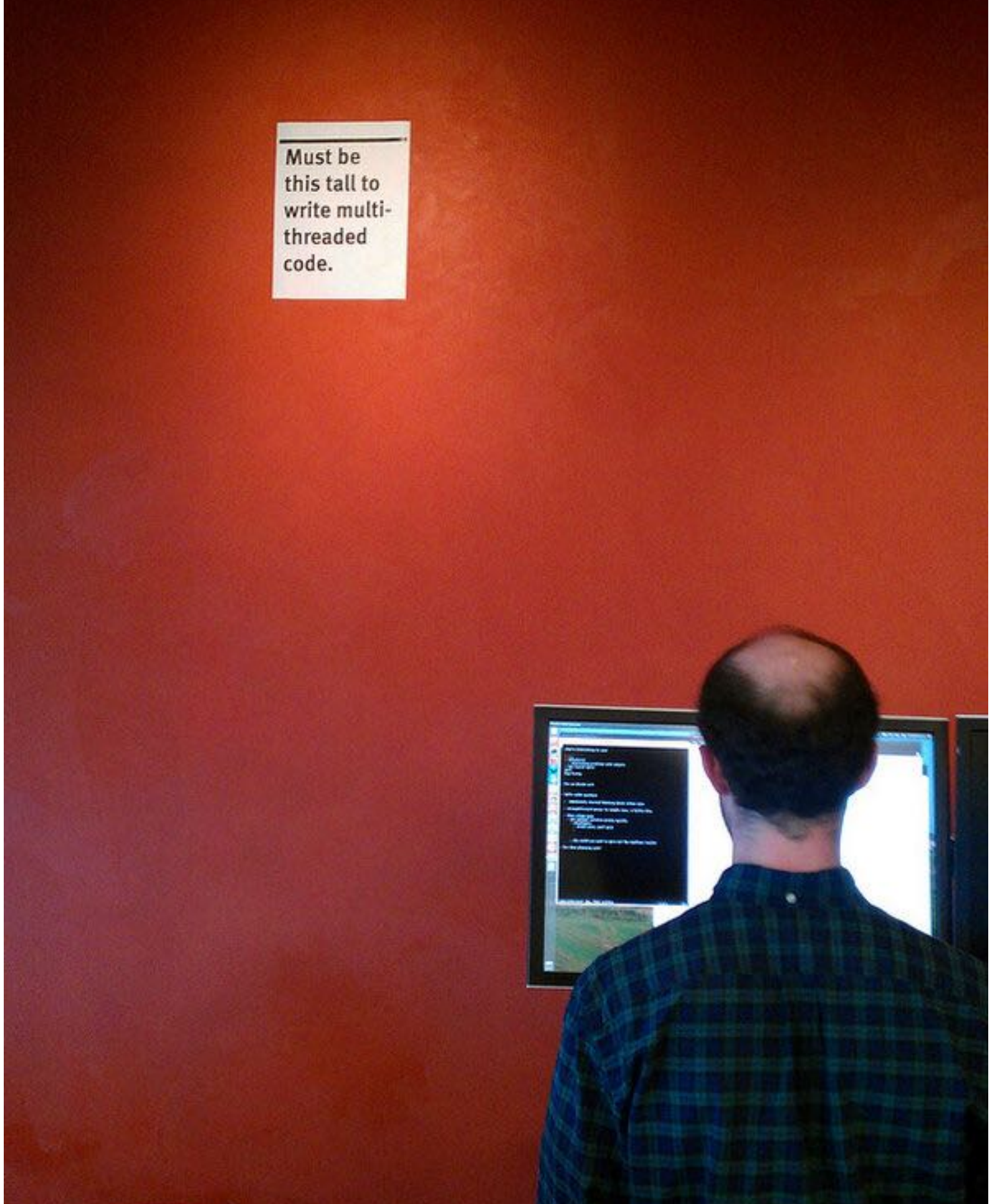
Threading Costs

- Extremely easy to get wrong
- Deadlocks
- Race conditions
- REALLY REALLY REALLY easy to get wrong


Must be
this tall to
write multi-
threaded
code.

threads/Trouble.java –>

threads/WhatWentWrong.java –>


Must be
this tall to
write multi-
threaded
code.

Homework 5

- Online at https://www.cs.uic.edu/~psnyder/cs342-summer2017/homework/hw5.html

- Examples of problem cases?