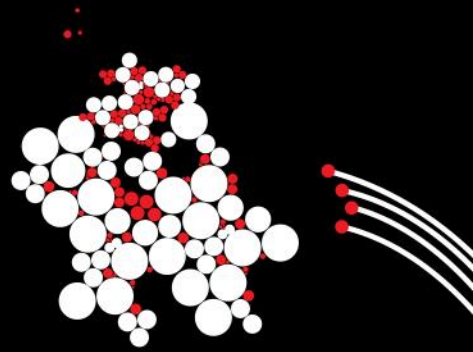
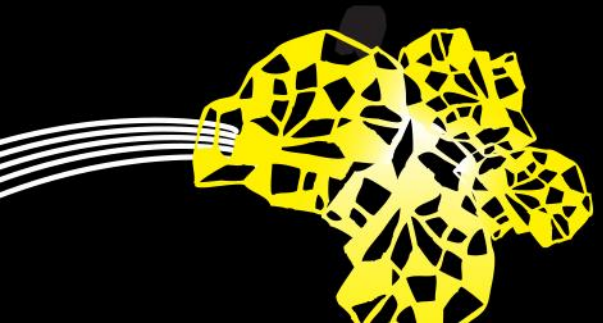
UNIVERSITY OF TWENTE.



Concurrency

Topic of Software Systems (TCS module 2)

Lecturer: Marieke Huisman



WHAT IS CONCURRENCY?

- Most of your programs so far: single-threaded
  - Only one execution of the code
  - Also: only one execution to think about

- Now: multi-threaded programs (also called concurrent or parallel)
  - As if multiple programs are running in the same “Java world”
  - Also: more difficult to reason about code correctness
    - Intentional *and* unintentional interaction between threads

- Actually: JVM already has extra threads, for example for garbage collection!

WHAT IS CONCURRENCY?

- Why? More efficient!
  - Computers have multiple processors (and processors have multiple cores)
    - Great for divide-and-conquer algorithms like mergesort!
  - Threads can be blocked
    - Waiting for disk or network I/O
    - Graphical user interfaces
  - Background operations should not block computations
    - Garbage collection
- Why? More natural!

- How do threads communicate?
  - Via shared memory and/or message passing and/or monitors

WHAT IS CONCURRENCY?

- Thinking about your multi-threaded program

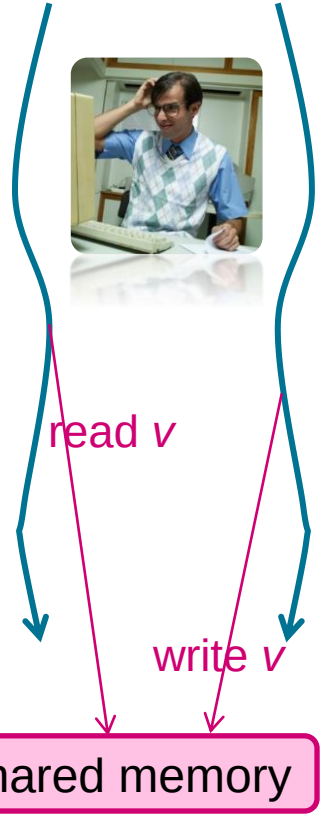
```
// A little experiment with a naive implementation of mergesort
// using the advanced Java work stealing thread pool to
// sort an ArrayList of random Integes

Available hardware threads: 6
to sort 10000000 integers with Java's default TimSort: 5.455 sec.
to sort 10000000 integers with parallel mergesort 1: 10.928 sec.
to sort 10000000 integers with parallel mergesort 2: 6.585 sec.
to sort 10000000 integers with parallel mergesort 4: 4.646 sec.
to sort 10000000 integers with parallel mergesort 8: 4.315 sec.
```

- Multiple “workers” (threads) collaborate on a result
- Mergesort: let other threads sort the smaller array concurrently, then merge sequentially
- User interface: some threads handle input, other threads handle output

CHALLENGES WITH CONCURRENCY

- Thinking about your multi-threaded program is hard
  - Especially if you really want your program to be fast and efficient
  - Concurrency bugs are hard to debug: Heissenbugs
- Two examples of concurrency-specific errors:
  - Accessing objects from multiple threads: data race / race condition
  - Multiple threads waiting for each other: deadlock
- Concurrency errors can be deadly
  - “Therac-25” race condition
  - North American Blackout of 2003



CONCURRENCY IN JAVA

More about concurrency: Programming Paradigms (Module 2.4)

- Thread handling
  - Thread creation: Implementing Runnable interface
  - Terminated threads can be joined
- Threads share data
- Access to data should often be synchronized to avoid data races
  - Every object is a lock
  - Synchronized code block
  - Synchronized methods
  - Lock interface
- Inter-thread communication about object state
  - Wait-notify
  - More fine-grained: use conditions

Extensive library for concurrency: java.util.concurrent