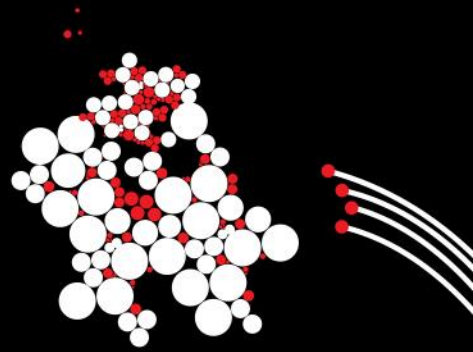
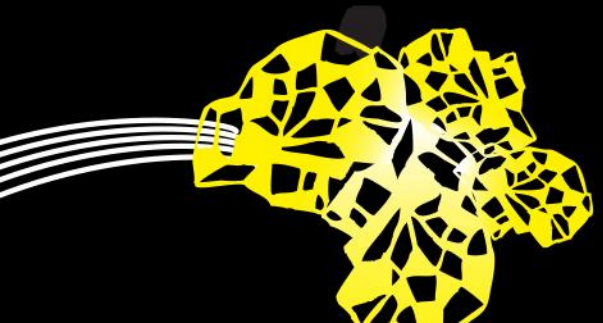
UNIVERSITY OF TWENTE.



Monitors

Topic of Software Systems (TCS module 2)

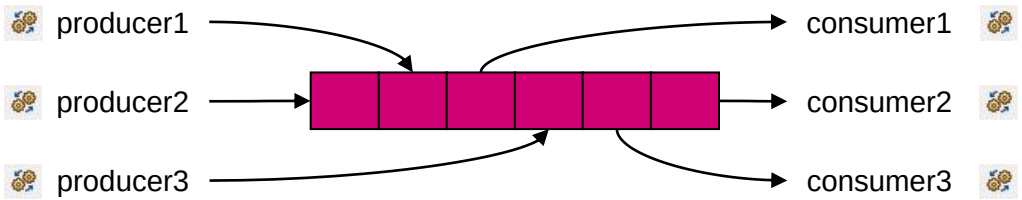
Lecturer: Marieke Huisman



WAITING FOR ANOTHER THREAD

Sometimes threads want to wait for a certain condition

Example: Producer and consumer threads with shared buffer



WAITING FOR A NON-EMPTY BUFFER

```
public class Consumer {
    ...
    // tries to read next element from buffer
    public Object getValue() {
        Object val = null;
        while (val == null) {
            synchronized (buffer) {
                if (!buffer.isEmpty()) {
                    val = buffer.getBuffer();
                }
            }
        }
        return val;
    }
}
```

Expensive ‘busy wait’ loop

COMMUNICATION BETWEEN THREADS

- Problem: How to let threads wait for certain conditions?

- Solution: conditions
  - A condition is associated to a lock
  - The wait or await operation waits until the condition is signalled
    - Release the lock, then suspend the thread
    - After the thread is woken up, reacquire the lock
  - The signal operation is called to indicate that the condition is now true
    - Typically (for example in Java) this wakes up 1 thread
  - The broadcast or signalAll operation wakes up all threads waiting for the condition
  - Signal-and-continue: the signaling thread continues after signal (behavior of Java)
  - Signal-and-wait: the signaling thread yields to the signaled thread

COMMUNICATION BETWEEN THREADS: MONITORS

- Java offers an easy method for doing this directly: monitors
- A monitor combines the functionality of a lock and a condition

- Every object in Java is also a monitor
  - `obj.wait()` releases mutex `obj` and wait
  - `obj.notify()` wakes up one arbitrary thread waiting for `obj`
  - `obj.notifyAll()` wakes up all threads waiting for `obj`

- These methods must only be called if a thread holds the lock

- `notifyAll`: more expensive
- `notify`: risk to wake up the 'wrong' thread

GETVALUE WITH WAIT-NOTIFY

```
public Object getValue() {
  synchronized (buffer) {
    while (buffer.isEmpty())
      try {
        buffer.wait();
      } catch (InterruptedException e) {
        ...
    }
    // read from the buffer
    buffer.notifyAll();
    return buffer.getBuffer();
  }
}
```

Wait for a producer to add an item to the buffer

Wake all threads waiting for an event in the buffer

MORE FINE-GRAINED WAIT-NOTIFY

- Condition interface, associated to Lock interface
  - await
  - signal
- Associate two conditions with buffer:
  - Condition empty
  - Condition full
- getValue:
  - empty.await();
  - full.signal() – to wake up producer threads

CONCURRENCY TAKE-HOME

More about concurrency: Programming Paradigms (Module 2.4)

- Threads
  - 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