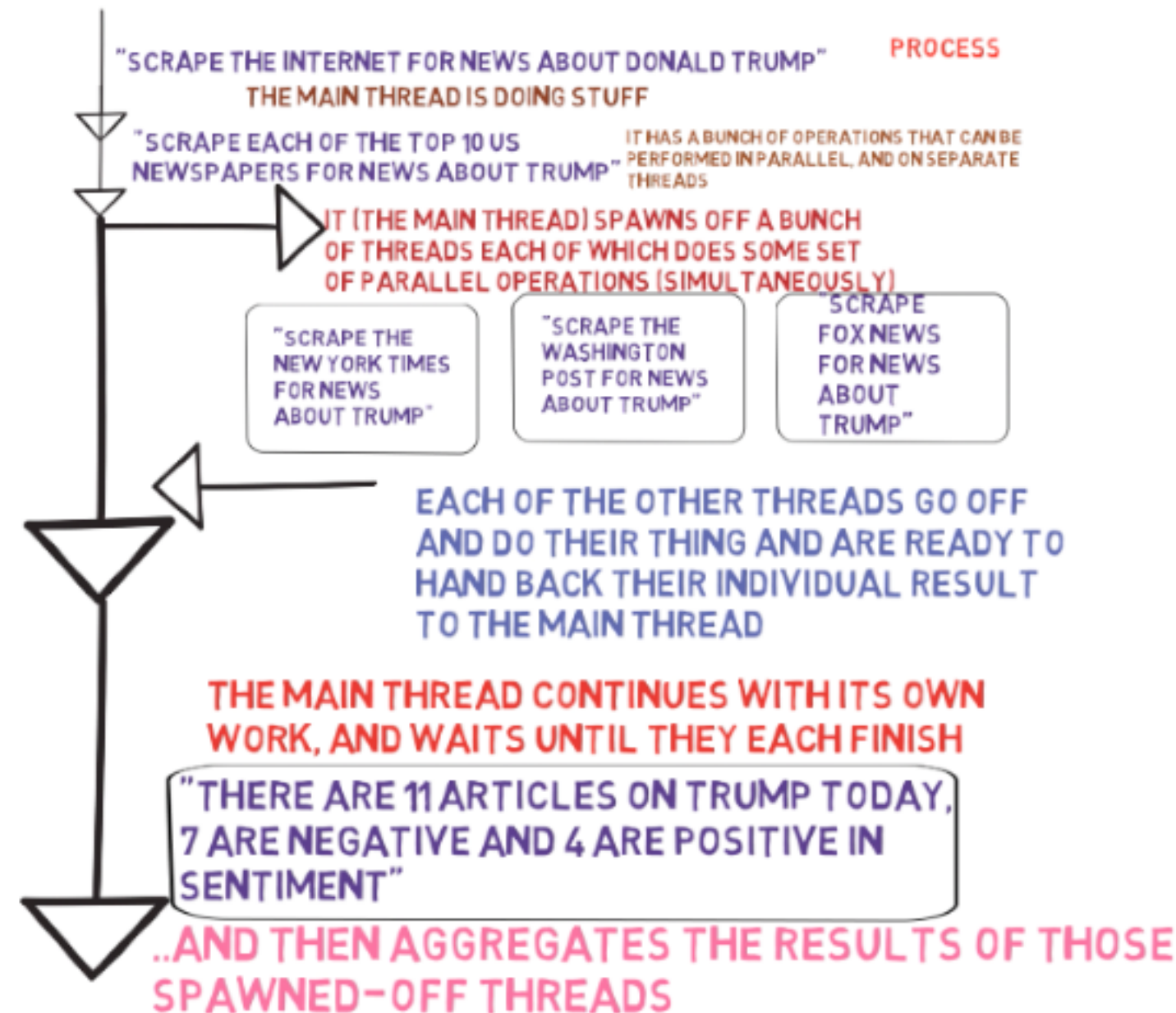


THE COMMAND PATTERN IN ACTION: THREADING

A TYPICAL USE-CASE FOR THREADING



IN JAVA THERE ARE 2 WAYS IN WHICH
THIS COULD BE ACCOMPLISHED

OLD-SCHOOL

RUNNABLE INTERFACE

IS IMPLEMENTED BY A CLASS WITH THE
OPERATIONS TO BE CARRIED OUT ON THE
OTHER THREADS

"THREAD" IN-BUILT CLASS

OBJECTS OF THE THREAD CLASS
TAKE IN THE RUNNABLE OBJECTS
AND RUN THEM ON INDIVIDUAL
THREADS

"THREAD.JOIN()" ON THE THREADS

THE MAIN CLASS CALLS THE
.JOIN METHOD ON EACH THREAD
WHICH WILL WAIT UNTIL THE
THREAD FINISHES

NEW SCHOOL

CALLABLE INTERFACE

IS IMPLEMENTED BY A CLASS WITH THE
OPERATIONS TO BE CARRIED OUT ON THE
OTHER THREADS

"EXECUTORS" IN-BUILT CLASS

JAVA PROVIDES HELPER OBJECTS THAT KNOW
HOW TO START, MANAGE AND STOP
'CALLABLE' OBJECTS

"FUTURE.GET()"

FUTURES ARE OBJECTS WHICH WILL HOLD
RESULTS IN THE FUTURE (I.E. ONCE THE
CALLABLE OBJECT FINISHES WHATEVER STUFF
IT HAD TO DO ON THE OTHER THREAD)

**BOTH THESE WAYS DIRECTLY MAKE
USE OF THE COMMAND PATTERN**

WE THEN ASK THAT THIS
OBJECT "DO ITS THING"
(I.E. EXECUTE ITS ACTION)
ON A SEPARATE THREAD

Callable Runnable

1. DEFINE THE COMMAND OBJECT

```
Runnable runnable = new Runnable() {  
    @Override  
    public void run() {  
        System.out.println("Hello");  
    }  
};
```

```
Thread thread = new Thread(runnable);  
thread.start();
```

2. HERE THE ACTION IS TO PRINT
TO SCREEN, SO THE RECEIVER
(WHAT GETS ACTED UPON) IS SIMPLY
THE SCREEN

3. THE INVOKER IS A THREAD
OBJECT - THE CLIENT CALLS
"START" ON THAT OBJECT TO
TRIGGER THE EXECUTION OF
THE ACTION