WEEK TWELVE

Acknowledgements: Slides created based off material provided by Dr. Michael Raymer and Dr. Travis Doom

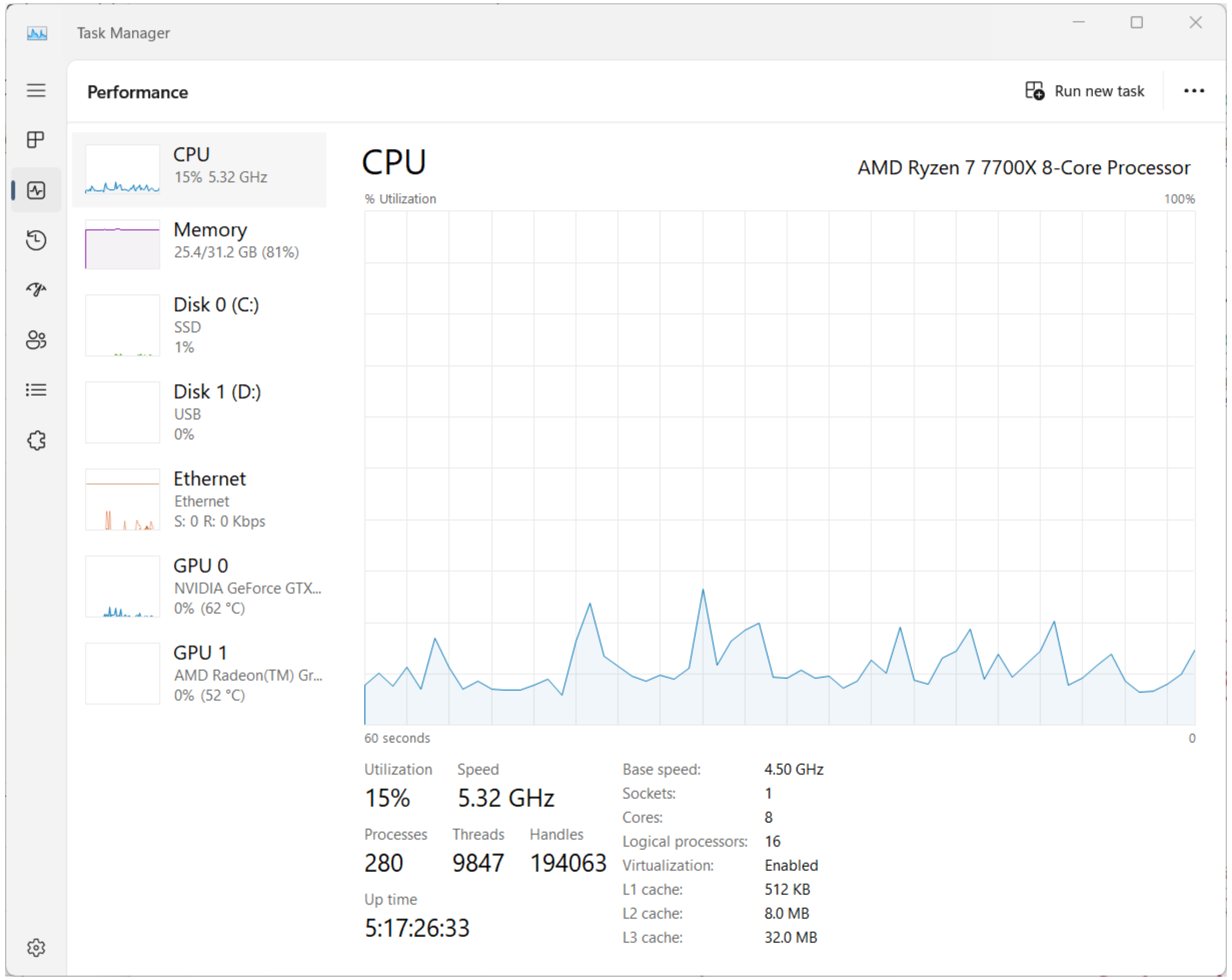
HOW DO COMPUTERS RUN MANY PROGRAMS AT THE SAME TIME?

Multiple cores

Pre-emptive multitasking

EXECUTION CORES

- Task manager shows number of cores on your CPU



AMD Ryzen 7 7700X 8-Core Processor

Utilization 15%
Speed 5.32 GHz
Processes 280
Threads 9847
Handles 194063
Up time 5:17:26:33

Base speed:	4.50 GHz
Sockets:	1
Cores:	8
Logical processors:	16
Virtualization:	Enabled
L1 cache:	512 KB
L2 cache:	8.0 MB
L3 cache:	32.0 MB

PROCESSES

- A process consists of:
  - Memory Space
    - Heap
    - Stack
    - Globals/Statics
    - Code
  - Program State
    - Program counter
    - Execution state: Running, waiting, sleeping, etc.

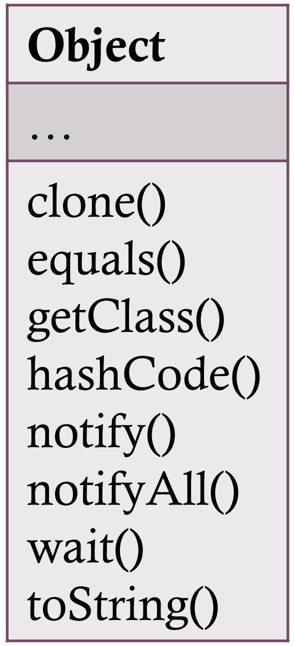
Each program executes as a single process.

THREADS

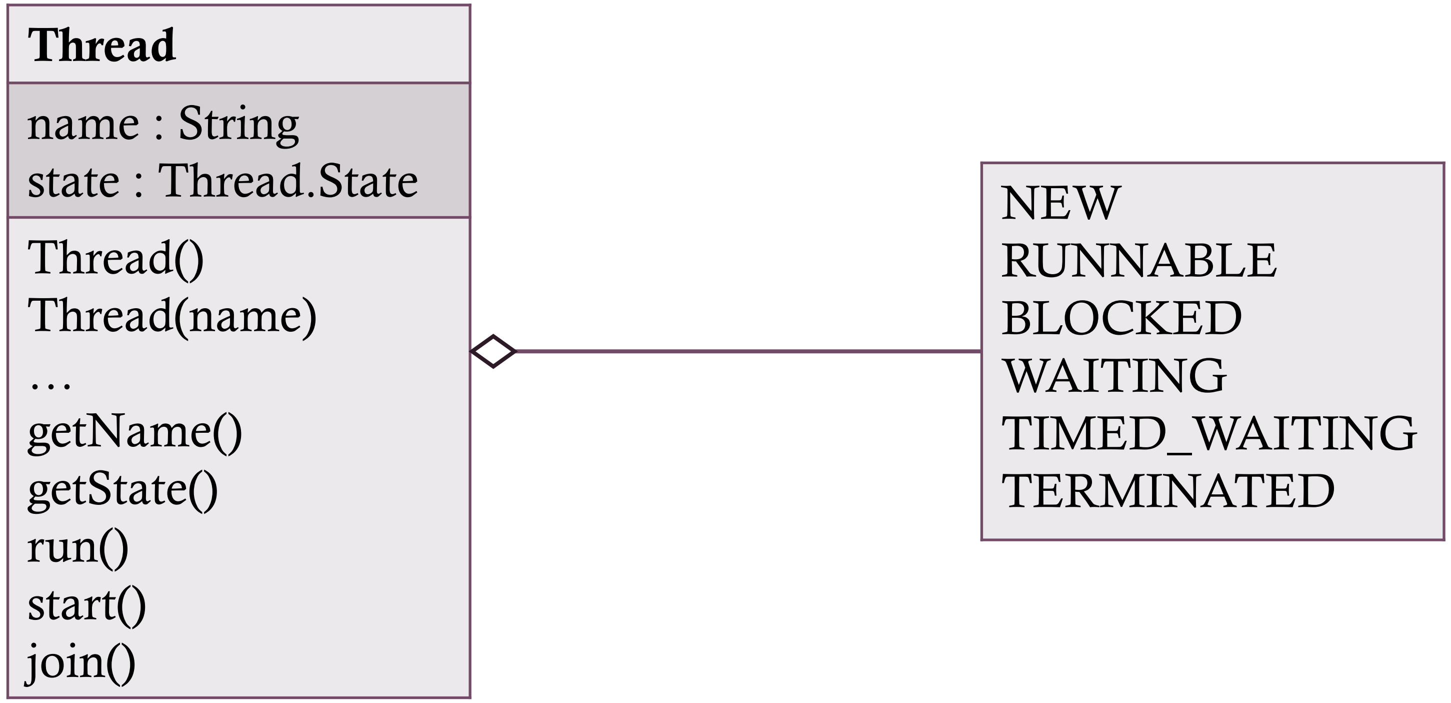
- An execution thread (aka a lightweight process) is a sequence of instructions being executed
- Threads share memory space but have their own stack frames
- A process can have multiple threads
- So far, we’ve only seen one or two execution threads running at a time

THREADS IN JAVA

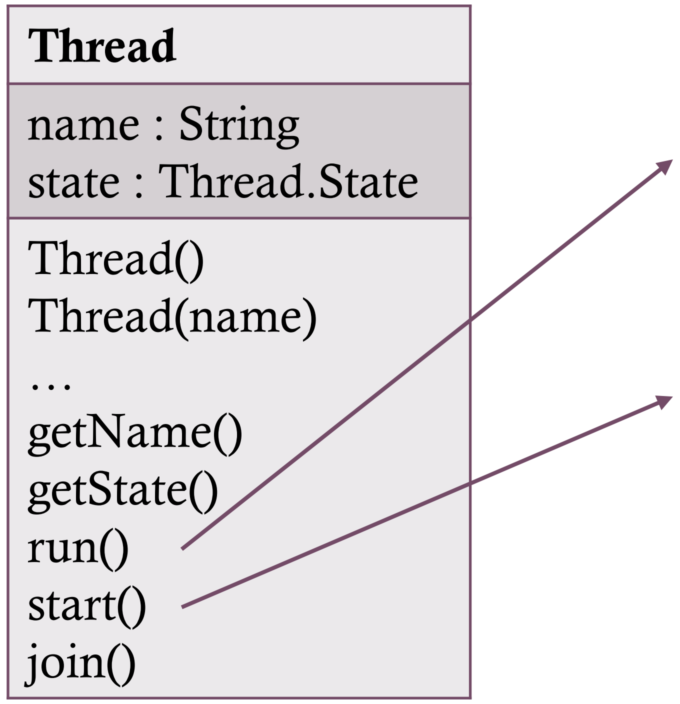
- Concurrency and threads are a core part of the java language (no import required!)



CLASS THREAD



CLASS THREAD



- run():
  - An ordinary method. Your code goes here!
- start():
  - Creates a new execution thread and starts executing run() in it.

RUN() VS START()

- Any code we want executed by a separate thread should go in the run() method
- When it’s time to generate the threads, call start()
- start() has code behind the scenes to create a new thread and execute the run() method

- Tips:
  - Do not override start(), only override run()
  - Do not call run(), unless you want everything to run on the same thread

LET’S TRY AN EXAMPLE

THREAD STATES

- **NEW:** No execution thread (start() hasn’t been called yet)
- **RUNNABLE:** Execution thread is running or waiting for the OS to run it
- **BLOCKED:** Waiting for a monitor lock
- **WAITING:** Waiting for another thread to wake this thread
- **TIMED_WAITING:** Waiting for a fixed amount of time to wake
- **TERMINATED:** Execution thread is finished

BASIC THREAD STATE PROGRESSION

```
public static void main(String[] args) {
        Thread t1 = new Thread(“thread1”);


        t1.start();

                         public void run(){
                              for (int i = 0; i < 20; i++){
                                   System.out.println(i);
                              }
                         }

}
```

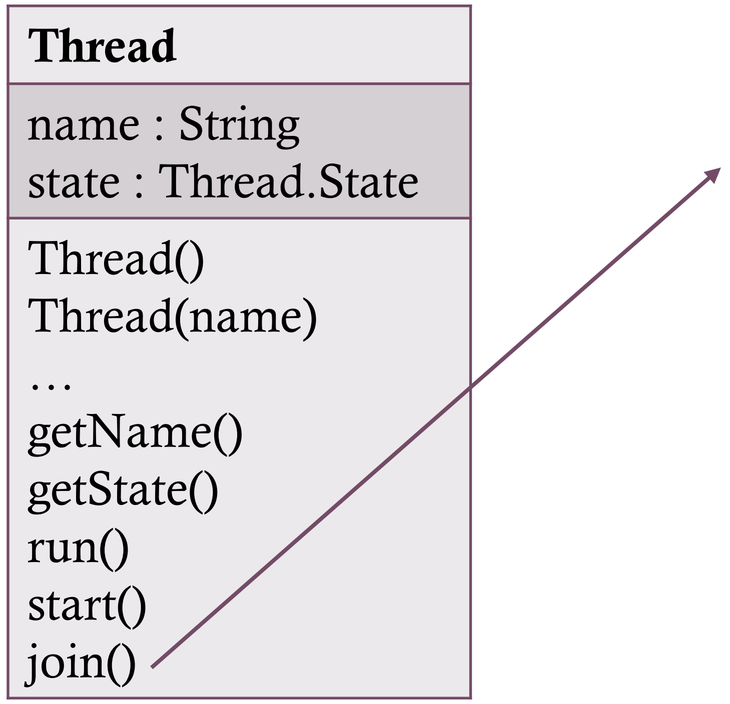
State:

NEW

RUNNABLE

TERMINATED

JOIN()



- join():
  - Wait for a thread to reach the TERMINATED state, then move on to the next instruction
  - Can be used when worker threads need to finish before the main thread continues

MULTITHREADING EXAMPLES

- When you want things running independently
  - For example, if you want an animation running while a file downloads
  - Games often require multiple threads for smooth play
  - Web server that handles multiple clients simultaneously
- When you have a lot of work to do
  - Divide up into parts and run one thread on each part
  - Protein folding problem