USF UNIVERSITY OF SAN FRANCISCO
CHANGE THE WORLD FROM HERE

Multithreading

CS 272 Software Development

Professor Sophie Engle
Department of Computer Science

Terminology

- **Process**
  - An instance of a program currently executing
  - Assigned its own resources and memory space
  - Contains at least one **thread of execution**
- **Thread**
  - Exists within a process and shares its resources
  - Similar to a **lightweight process**

http://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

Terminology

- **Concurrency**
  - Performing more than one action simultaneously
  - May be applied to processes or threads
- **Multithreading**
  - Running multiple threads per process
  - Create **worker threads** to handle specific tasks

http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

Multithreading

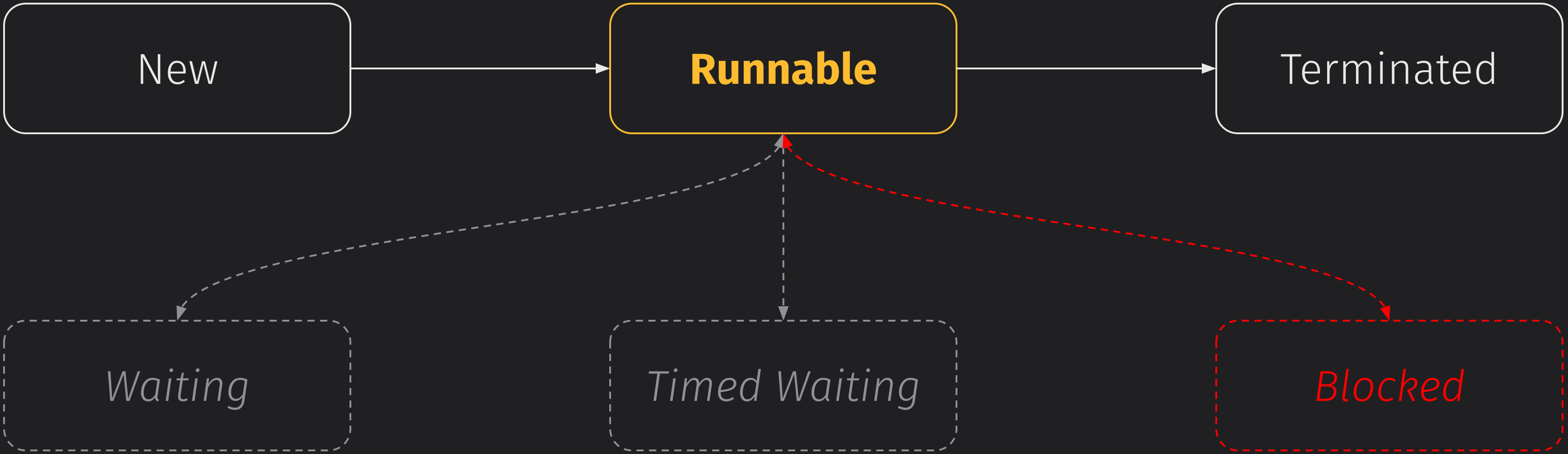
- Start with a **large** and **parallelizable** problem
  - i.e. can break a large problem into smaller tasks that can be completed simultaneously
- Create **worker threads** to handle smaller tasks
- Use **synchronization** to get final results from workers

Thread Lifecycle

- Create a **new** thread and initialize members
  - Once complete, thread becomes **runnable**
- A **runnable** thread is ready to perform work
  - Might be **waiting** for something, or be **blocked** from a resource that is busy
- When work is complete, thread is **terminated**
  - Data members still around in memory

https://developer.ibm.com/tutorials/j-threads/#a-thread-s-life

Thread States



https://www.cs.usfca.edu/~cs272/javadoc/api/java.base/java/lang/Thread.State.html

Multithreading Classes

- `Object` Class
  - `notify()`, `notifyAll()`, `wait()`
- `Runnable` Interface
  - `run()`
- `Thread` Class
  - `start()`, `join()`, `sleep()`, and others

https://www.cs.usfca.edu/~cs272/javadoc/api/java.base/java/lang/Thread.html

Multithreading in Java

- Creating Threads
  - Extend `Thread` and override `run()`
  - Implement `Runnable`, pass to `Thread` constructor
- Managing Threads
  - Manually (call `start()`, `join()`, etc. in code)
  - Via a task executor (discussed later)

http://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html

Obstacles

- Creating threads requires **time** and **resources**
  - For small amounts of work, *may* slow down code
  - For large amounts of work, *may* speed up code
- Must **synchronize** access to **shared data**
- Order of operations is **non-deterministic**
  - Difficult to debug and replicate problems

USF UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Software Development
Department of Computer Science

Professor Sophie Engle
sjengle.cs.usfca.edu