Thread

Process: simply an executing program - it can be composed of one to many threads

Thread: set of commands to be executed typically divided into as small of chunks as possible.

- All Java programs have at least one thread
    - called the main thread
- Threads can spawn off other threads
- We can create threads beyond the main thread

Multi-Threading

- a concept where we have the ability for multiple threads to exist concurrently
    - more than one thread that runs individually in parallel
- It's important to note that multithreading does not exist in every programming language. JavaScript is a single-threaded language.
- Asynchronous != Multithreading

Why?

- In general, two heads are better than one.
- It allows us to conserve some of the wasted downtime in a thread or a process
    - it will eventually it will break down (when you have more threads than processors)

TERMS

Scheduler: responsible for deciding what thread runs and when.

synchronized keyword: only one thread can access this resources (method, field) at a time - 'mutual exclusion'

Race Condition = when you have two or more threads running, there is the potential that those two or more threads will need access to the same resource at the same.

Deadlock: when two or more threads are trying to access resources that the other thread is using.

- nothing gets done and no resources are moving.

LiveLock: two or threads are needing two or more resources, and they're continually swapping.

- nothing actually gets accomplished, but resources are moving around all over the place

Thread Lifecycle

- Different phases that a Thread goes through

NEW => a thread's lifecycle begins. Created, but not yet started.

RUNNABLE => Thread has been started, it's in a ready state, and waiting for a resource or waiting for the Scheduler to tell it run. `start()`

RUNNING => When the Scheduler has selected this thread to run. `run()`

BLOCKED / WAITING => This can happen during the running phase - waiting for the monitor to release a lock on a synchronized resource.

TERMINATED => the task is complete and the thread has finished.

Created Threads in Java

- Create a class that `extends` the Thread Class
    - Override the `run()` method.
    - Pass an instance of this class into the Thread Constructor
    - call the `start()` method
- Create a class that `implements` the Runnable Interface
    - Implement the `run()` method
    - Create an object of the class
    - call the `start()`