Threads

Threads

Threads are how we can get more than one thing to happen at once in a program.
- Making your program multitask!

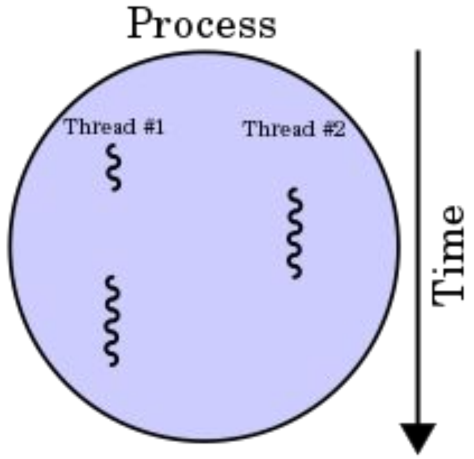
A Thread is just a **sequence of instructions** to execute

Program Vs Thread

Sequential programs have a beginning, execution sequence and end. At any given time during runtime, there is a single point of execution.

Threads contain those same qualities as a sequential program. However, a thread itself is not a program and cannot run on its own.

Threads run with in a program and share memory



Creating a Thread in Java

```
class Student extends Thread{
    public void run(){
        // listen , talk , fidget
        ...
    }
}
    ...
Student myStudent = new Student();
myStudent.start() ;
```

```
class Student implements Runnable{
    public void run(){
        // listen , talk , fidget
        ...
    }
}
...
Student myStudent = new Student();
Thread myThread = new Thread(myStudent)
myThread.start();
```

Controlling Threads

- start() and stop() - start and stops thread
- sleep() - will put thread to sleep
- join() - causes parent thread to wait for thread to die before continuing
- interrupt() - will wake up thread that is sleeping or blocked

Thread Life Cycle

Threads continue until one of the following

- done executing their assigned task
- Interrupted by an uncaught exception
- stop() method is called

What happens if the run() method never terminates, and the application that started the thread never calls the stop() method?

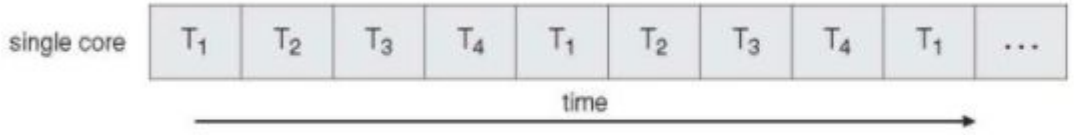
The thread remains alive even after the application has finished! (so the Java interpreter has to keep on running...)

Parallel Execution

Number of threads that can execute in parallel depends on the hardware of your machine.

# CPU cores= # of Threads that can truly run concurrently.

Threads will If take turns executing if you have more threads than running than CPU Cores



OS can time slice between the four Threads T1...T4