Threads

Multithreaded programs

Synchronization

Threads

- Why do we need parallel programming?
  - Multiple tasks at once
- What is a process
  - Self-contained execution environment
  - Private run-time resources and own memory space
- What is a thread
  - Lightweight process (fewer resources)
  - Exists within a process (every process has at least one)
  - Share process's resources

Thread Sleep

- Sleeping thread for a specified time
- Static method!
- Causes ONLY THE CURRENT thread to sleep

```java
for (int i = 1; i <= 10; i++) {
        Thread.sleep(200);
        System.out.println("Counting... " + i);
}
```

Runnable

```
public class MyRunnable implements Runnable {

    @Override
    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println("Method run " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Thread

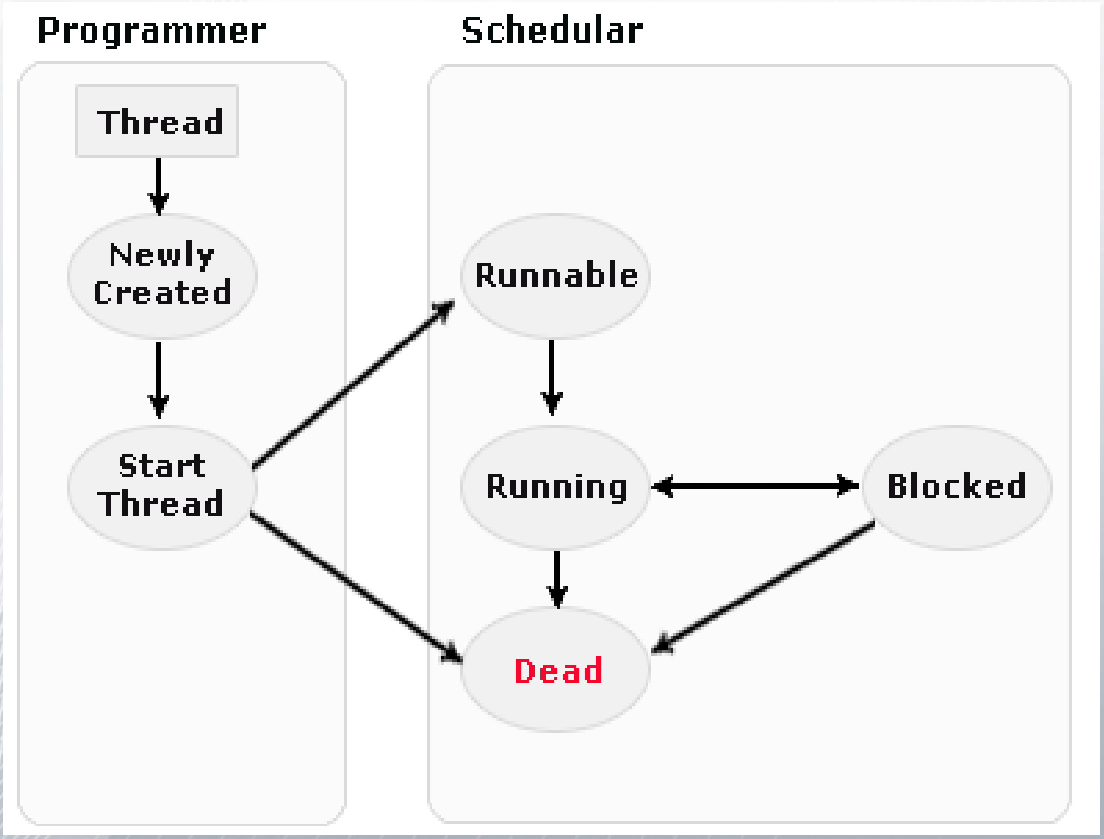
```
public class MyThread extends Thread {
    @Override
    public void run() {
        for (int i = 1; i <= 10; i++) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                return;
            }
            System.out.println("counting... " + i);
        }
    }

    public MyThread(String name) {
        super(name);
    }
}
```

Exercise

- Create a Runnable which counts to 5 and prints the numbers to the console
- Make 5 threads using it and start them
- Slow down the counting (sleep)
  - Run the program several times. Which thread prints the first number?

Livecycle

- Buy invoking start() method it doesn't mean that the thread has access to the CPU
- Thread might not start right away.



Thread States

- New state – only instance of the thread is created and the actual thread is not alive
  - start() is not invoked yet
- Runnable (Ready to run) – just after the start() method
  - Waiting for a turn on the processor
- Running state – currently executing
  - Scheduler selects the thread from a runnable pool.

Thread States

- Blocked state
  - Enters in blocked state when waits for resources that are hold by another thread.
- Dead state – means the thread can't run ever again
  - A thread can be considered as “dead” when its run() method completes

More About Threads

- Every main() method starts a thread
  - All our previous programs were single-threaded programs
- Every thread has its own call-stack
- Every thread can be debugged as a separate program
- interrupt() vs stop() methods
  - What is a deprecated method?

Thread interrupt

```java
MyRunnable runnable = new MyRunnable();
Thread t1 = new Thread(runnable, "Thread 1");
Thread t2 = new Thread(runnable, "Thread 2");
Thread t3 = new Thread(runnable, "Thread 3");
Thread t4 = new Thread(runnable, "Thread 4");
Thread t5 = new Thread(runnable, "Thread 5");

t1.start();
t2.start();
t3.start();
t4.start();
t5.start();

Thread.sleep(2700);
t1.interrupt();
```

IsAlive()?

```
MyRunnable runnable = new MyRunnable();
Thread t1 = new Thread(runnable, "TestThread 1");
Thread t2 = new Thread(runnable, "TestThread 2");
Thread t3 = new Thread(runnable, "TestThread 3");
t1.start();
t2.start();
t3.start();

int i=0;
while(t1.isAlive() || t2.isAlive() || t3.isAlive()) {
   //Thread.sleep(50);
   i++;
}

System.out.println("All threads are dead");
System.out.println(i);
```

Thread Join?

- Current thread waits the joined thread

```
MyRunnable runnable = new MyRunnable();
Thread t1 = new Thread(runnable, "Thread 1");
Thread t2 = new Thread(runnable, "Thread 2");
Thread t3 = new Thread(runnable, "Thread 3");

t1.start();
t2.start();
t3.start();

t1.join();
t2.join();
t3.join();
System.out.println("All thread are dead...");
```

Synchronization

- Resources for a single-threaded application
- Shared resources between many threads
- Dead lock
  - Two or more threads are blocked forever waiting for each-other
  - Eclipse demo
- Bank and Bank account example
  - Eclipse demo
- Producer – Consumer
  - Eclipse demo

Let's code!

Simple bouncing ball game