

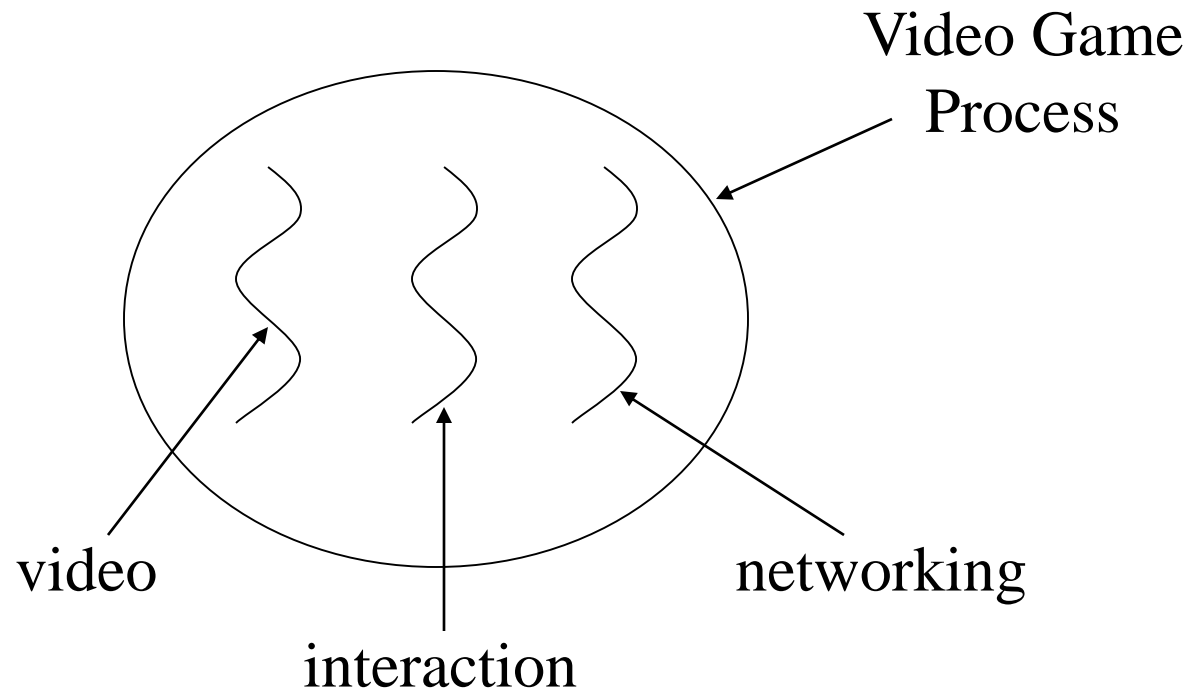
JAVA THREADS

SHAILESH S
ASST. PROFESSOR
DEPT. OF COMPUTER SCIENCE
SH COLLEGE

WHAT IS A THREAD?

- **Individual and separate unit of execution that is part of a process**
 - multiple threads can work together to accomplish a common goal
- **Video Game example**
 - one thread for graphics
 - one thread for user interaction
 - one thread for networking

WHAT IS A THREAD?



ADVANTAGES

- **easier to program**
 - 1 thread per task
- **can provide better performance**
 - thread only runs when needed
 - no polling to decide what to do
- **multiple threads can share resources**
- **utilize multiple processors if available**

DISADVANTAGE

- **multiple threads can lead to deadlock**
 - much more on this later
- **overhead of switching between threads**

CREATING THREADS (METHOD 1)

- **extending the Thread class**

- must implement the *run()* method

- thread ends when *run()* method finishes

- call *.start()* to get the thread ready to run

CREATING THREADS EXAMPLE 1

```
class Output extends Thread {
    private String toSay;
    public Output(String st) {
        toSay = st;
    }
    public void run() {
        try {
            for(;;) {
                System.out.println(toSay);
                sleep(1000);
            }
        } catch(InterruptedException e) {
            System.out.println(e);
        }
    }
}
```

EXAMPLE 1 (CONTINUED)

```
class Program {  
    public static void main(String [] args) {  
        Output thr1 = new Output("Hello");  
        Output thr2 = new Output("There");  
        thr1.start();  
        thr2.start();  
    }  
}
```

main thread is just another thread (happens to start first)
main thread can end before the others do
any thread can spawn more threads

CREATING THREADS (METHOD 2)

- **implementing Runnable interface**
 - virtually identical to extending Thread class
 - must still define the *run()* method
 - setting up the threads is slightly different

CREATING THREADS EXAMPLE 2

```
class Output implements Runnable {
    private String toSay;
    public Output(String st) {
        toSay = st;
    }
    public void run() {
        try {
            for(;;) {
                System.out.println(toSay);
                Thread.sleep(1000);
            }
        } catch(InterruptedException e) {
            System.out.println(e);
        }
    }
}
```

EXAMPLE 2 (CONTINUED)

```
class Program {  
    public static void main(String [] args) {  
        Output out1 = new Output("Hello");  
        Output out2 = new Output("There");  
        Thread thr1 = new Thread(out1);  
        Thread thr2 = new Thread(out2);  
        thr1.start();  
        thr2.start();  
    }  
}
```

main is a bit more complex
everything else identical for the most part

ADVANTAGE OF USING RUNNABLE

- remember - can only extend one class
- implementing runnable allows class to extend something else

CONTROLLING JAVA THREADS

`_.start()`: begins a thread running

`wait()` and **`notify()`**: for synchronization

- more on this later

`_.stop()`: kills a specific thread (deprecated)

`_.suspend()` and **`resume()`**: deprecated

`_.join()`: wait for specific thread to finish

`_.setPriority()`: 0 to 10 (MIN_PRIORITY to MAX_PRIORITY); 5 is default (NORM_PRIORITY)

JAVA THREAD SCHEDULING

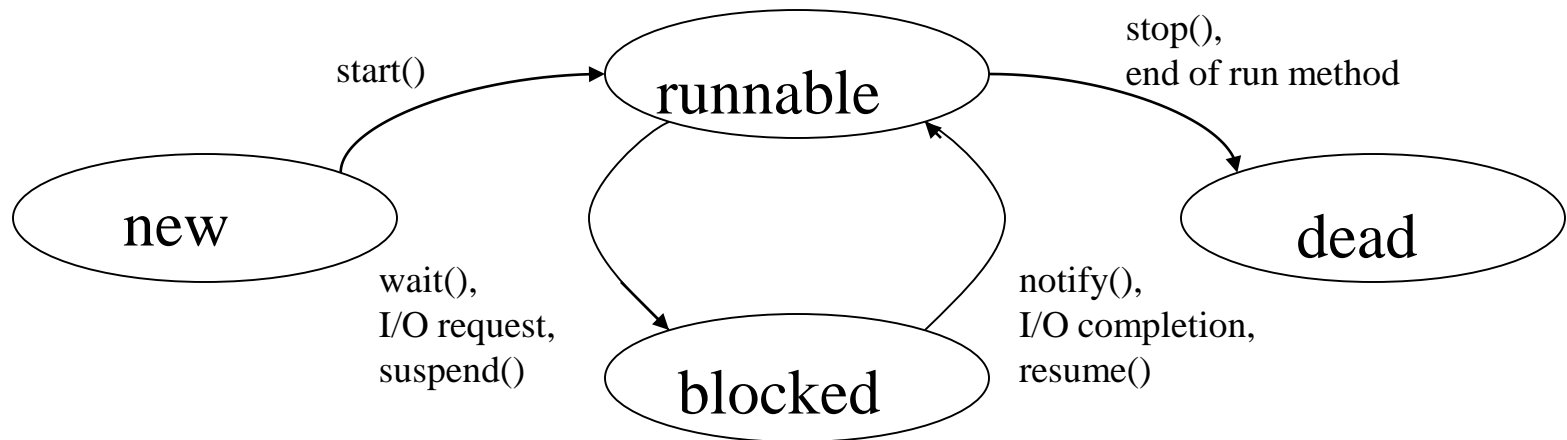
- **highest priority thread runs**
 - if more than one, arbitrary
- ***yield()*: current thread gives up processor so another of equal priority can run**
 - if none of equal priority, it runs again
- ***sleep(msec)*: stop executing for set time**
 - lower priority thread can run

STATES OF JAVA THREADS

➤ 4 separate states

- new: just created but not started
- runnable: created, started, and able to run
- blocked: created and started but unable to run because it is waiting for some event to occur
- dead: thread has finished or been stopped

STATES OF JAVA THREADS



JAVA THREAD EXAMPLE 1

```
class Job implements Runnable {
    private static Thread [] jobs = new Thread[4];
    private int threadID;
    public Job(int ID) {
        threadID = ID;
    }
    public void run() { do something }
    public static void main(String [] args) {
        for(int i=0; i<jobs.length; i++) {
            jobs[i] = new Thread(new Job(i));
            jobs[i].start();
        }
        try {
            for(int i=0; i<jobs.length; i++) {
                jobs[i].join();
            }
        } catch(InterruptedException e) { System.out.println(e); }
    }
}
```

JAVA THREAD EXAMPLE 2

```
class Schedule implements Runnable {
    private static Thread [] jobs = new Thread[4];
    private int threadID;
    public Schedule(int ID) {
        threadID = ID;
    }
    public void run() { do something }
    public static void main(String [] args) {
        int nextThread = 0;
        setPriority(Thread.MAX_PRIORITY);
        for(int i=0; i<jobs.length; i++) {
            jobs[i] = new Thread(new Job(i));
            jobs[i].setPriority(Thread.MIN_PRIORITY);
            jobs[i].start();
        }
        try {
            for(;;) {
                jobs[nextThread].setPriority(Thread.NORM_PRIORITY);
                Thread.sleep(1000);
                jobs[nextThread].setPriority(Thread.MIN_PRIORITY);
                nextThread = (nextThread + 1) % jobs.length;
            }
        } catch (InterruptedException e) { System.out.println(e); }
    }
}
```