Concurrent Computing

CSCI 201

Principles of Software Development

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

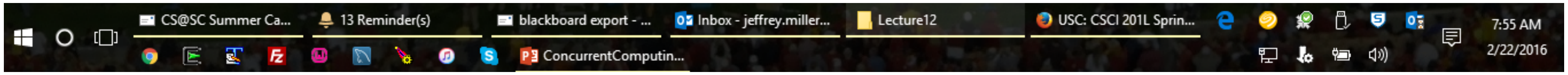
USC Viterbi
School of Engineering

Outline

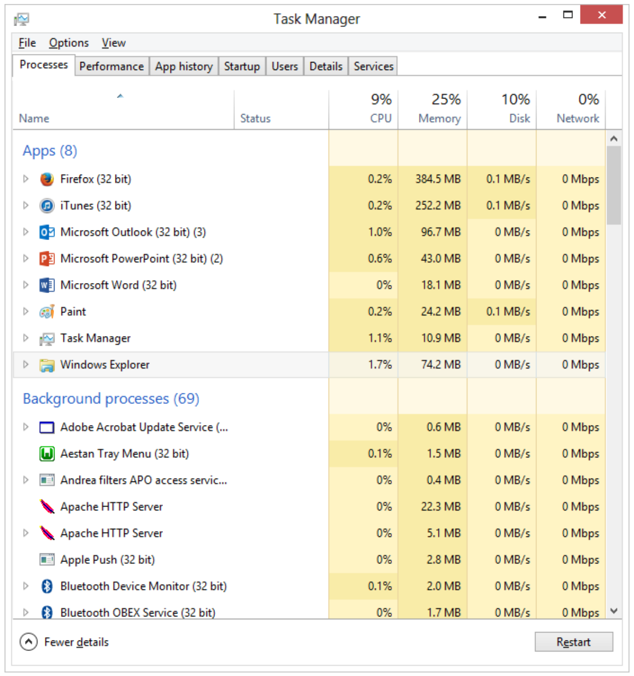
- Threads
- Multi-Threaded Code
- CPU Scheduling
- Program

Thread Overview

- Looking at the Windows taskbar, you will notice more than one process running concurrently



- Looking at the Windows task manager, you'll see a number of additional processes running
- But how many processes are actually running at any one time?
  - Multiple processors, multiple cores, and hyper-threading will determine the actual number of processes running

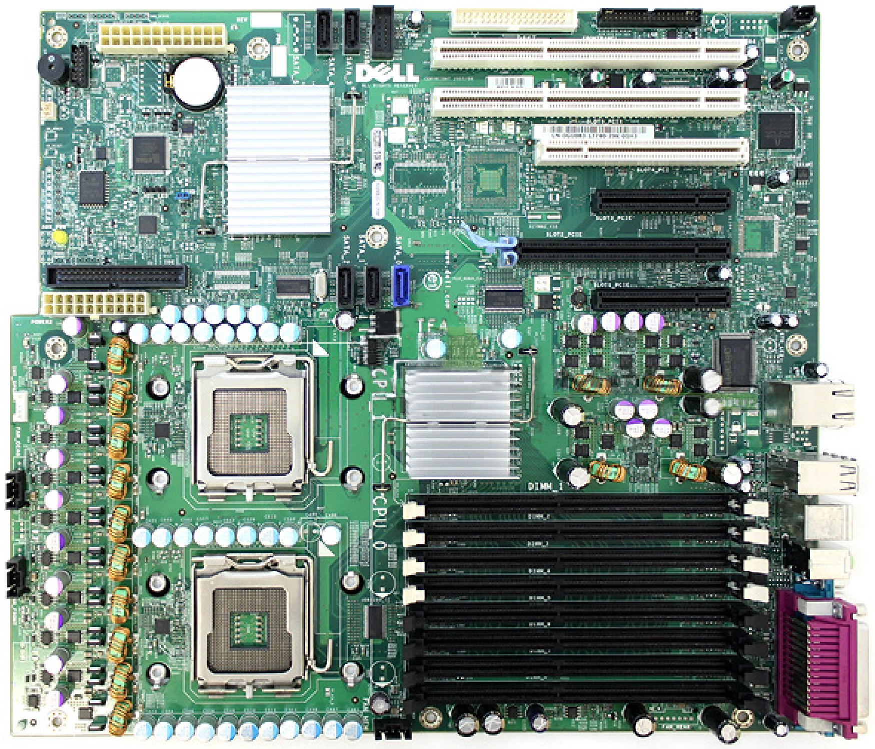

Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
Restart

Name	Status	9% CPU	25% Memory	10% Disk	0% Network
Apps (8)					
Firefox (32 bit)		0.2%	384.5 MB	0.1 MB/s	0 Mbps
iTunes (32 bit)		0.2%	252.2 MB	0.1 MB/s	0 Mbps
Microsoft Outlook (32 bit) (3)		1.0%	96.7 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (32 bit) (2)		0.6%	43.0 MB	0 MB/s	0 Mbps
Microsoft Word (32 bit)		0%	18.1 MB	0 MB/s	0 Mbps
Paint		0.2%	24.2 MB	0.1 MB/s	0 Mbps
Task Manager		1.1%	10.9 MB	0 MB/s	0 Mbps
Windows Explorer		1.7%	74.2 MB	0 MB/s	0 Mbps
Background processes (69)					
Adobe Acrobat Update Service (...		0%	0.6 MB	0 MB/s	0 Mbps
Aestan Tray Menu (32 bit)		0.1%	1.5 MB	0 MB/s	0 Mbps
Andrea filters APO access servic...		0%	0.4 MB	0 MB/s	0 Mbps
Apache HTTP Server		0%	22.3 MB	0 MB/s	0 Mbps
Apache HTTP Server		0%	5.1 MB	0 MB/s	0 Mbps
Apple Push (32 bit)		0%	2.8 MB	0 MB/s	0 Mbps
Bluetooth Device Monitor (32 bit)		0.1%	2.0 MB	0 MB/s	0 Mbps
Bluetooth OBEX Service (32 bit)		0%	1.7 MB	0 MB/s	0 Mbps

USC Viterbi
School of Engineering

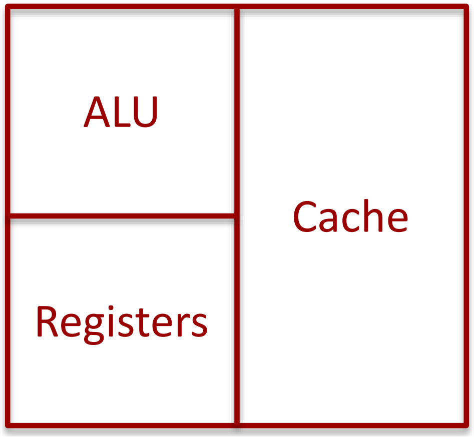
Redundant Hardware

Multiple Processors

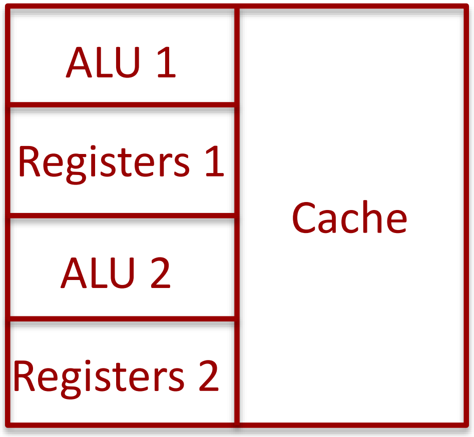


Hyper-threading has firmware logic that appears to the OS to contain more than one core when in fact only one physical core exists

Single-Core CPU



Multiple Cores



USC Viterbi School of Engineering

Programs vs Processes vs Threads

- Programs
  - An executable file residing in memory, typically in secondary storage
- Processes
  - An executing instance of a program
  - Sometimes this is referred to as a Task
- Threads
  - Any section of code within a process that can execute at what appears to be *simultaneously*
  - Shares the same resources allotted to the process by the OS kernel

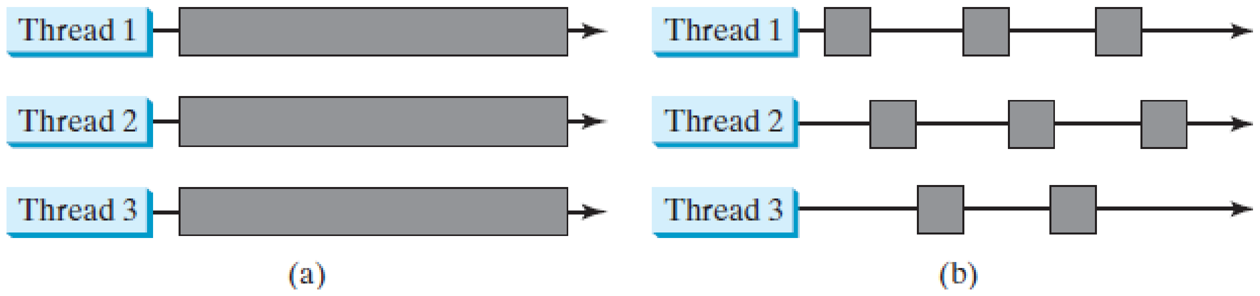


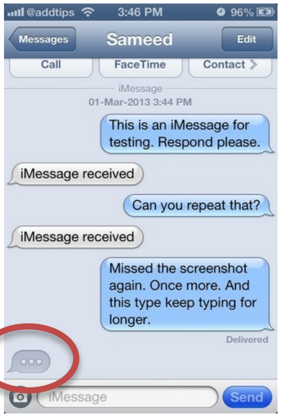
FIGURE 29.1 (a) Here multiple threads are running on multiple CPUs. (b) Here multiple threads share a single CPU.

Concurrent Programming

- Multi-Threaded Programming
  - Multiple sections of a process or multiple processes executing at what appears to be simultaneously in a single core
  - **The purpose of multi-threaded programming is to add functionality to your program**
- Parallel Programming
  - Multiple sections of a process or multiple processes executing simultaneously in different cores or different CPUs with a shared memory space
  - **The purpose of parallel programming is to speed up execution of your program by using redundant resources *typically* on your computer**
- Distributed Programming
  - Multiple sections of a process or multiple processes executing on multiple CPUs with different memory spaces (which could be located on different computers)
  - **The purpose of distributed programming is to speed up execution of your program by using resources *typically* on other computers**

Examples of Concurrent Programming

- Chatting
  - › While being able to type and send a chat message, the program can also receive a chat message
- Message Updates
  - › In social media applications, while you are reading other posts, an update can come in to notify you that a new post has been made
- Multi-player Games
  - › There are many different actions that are happening with different characters in a multi-player game. Each character could be operating in its own thread



iMessage
01-Mar-2013 3:44 PM

This is an iMessage for testing. Respond please.

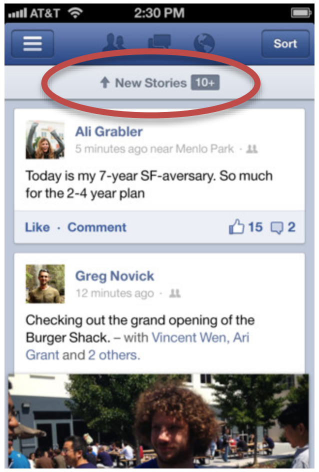
iMessage received

Can you repeat that?

iMessage received

Missed the screenshot again. Once more. And this type keep typing for longer.

Delivered



Ali Grabler
5 minutes ago near Menlo Park

Today is my 7-year SF-aversary. So much for the 2-4 year plan

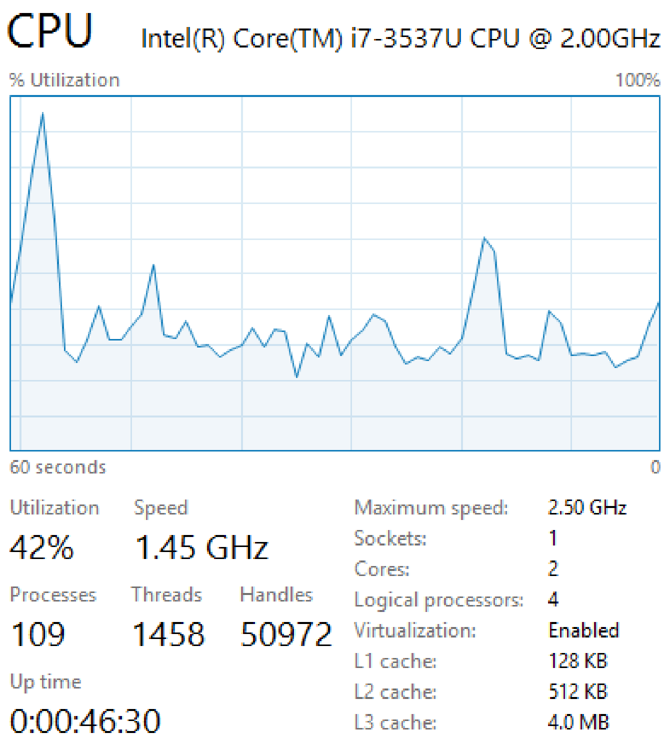
Like · Comment 15 2

Greg Novick
12 minutes ago

Checking out the grand opening of the Burger Shack. – with Vincent Wen, Ari Grant and 2 others.

Thread Details

- The operating system will switch processes and native threads into and out of the CPU
  - This is called time slicing
  - NOTE: In Java, the **JVM** switches threads into and out of the CPU, and the **OS** switches the **JVM** into and out of the CPU
- Time slicing is efficient because the CPU sits idle a lot of the time
  - For example, waiting for user input or waiting on another process
- All Java programs have two threads
  - The thread that contains the `main` method
  - The garbage collector

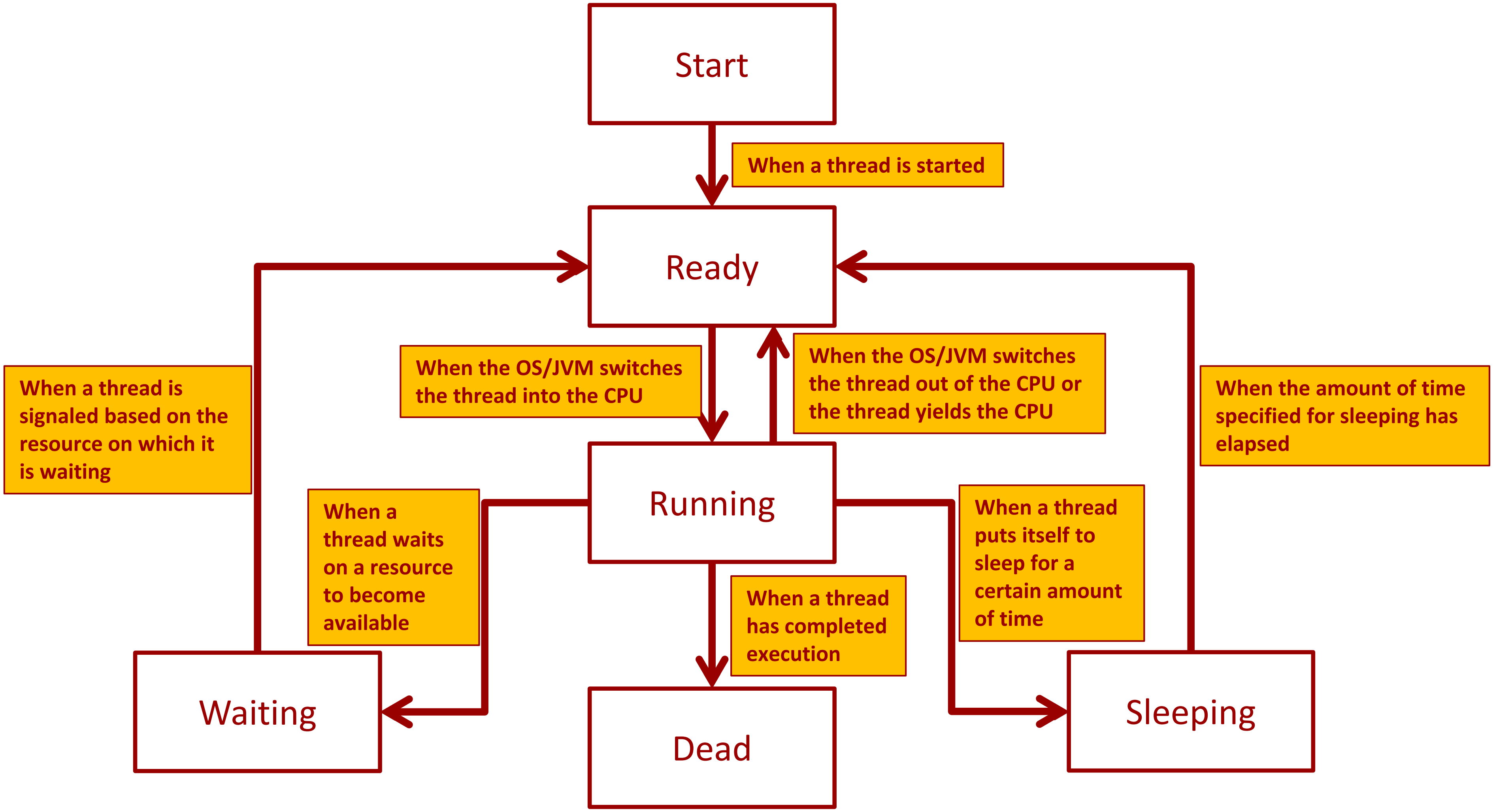


Utilization	Speed	
42%	1.45 GHz	
Processes	Threads	Handles
109	1458	50972
Up time		
0:00:46:30		

Maximum speed:	2.50 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	4.0 MB

USC Viterbi School of Engineering

Thread States


Start
When a thread is started
Ready
When a thread is signaled based on the resource on which it is waiting
When the OS/JVM switches the thread into the CPU
When the OS/JVM switches the thread out of the CPU or the thread yields the CPU
When the amount of time specified for sleeping has elapsed
Running
When a thread waits on a resource to become available
When a thread puts itself to sleep for a certain amount of time
When a thread has completed execution
Waiting
Dead
Sleeping

Multi-Threaded Debugging

- Since we have no control over when a thread will be switched into and out of the CPU, there are many different executions of a program
- If you see an error during one execution, you may not see the same error during the next execution
  - This does not mean that the code is not deterministic, but rather that there are many different paths of execution

USC Viterbi
School of Engineering

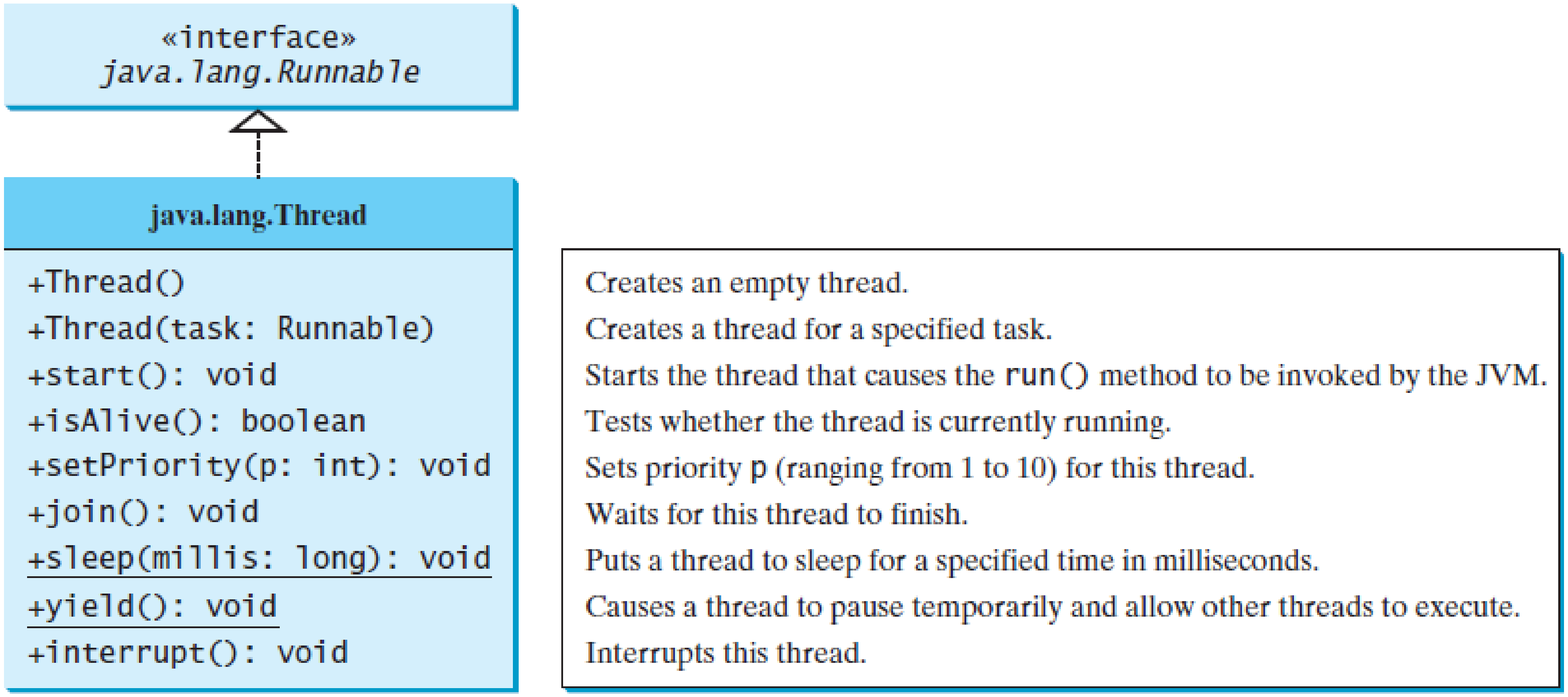
Outline

- Threads
- Multi-Threaded Code
- CPU Scheduling
- Program

Creating a Thread

- There are two ways to create multi-threaded code in Java
  - Implement the `Runnable` interface, pass it into an instance of a `Thread`, and start the thread
    - `Runnable` is in `java.lang`, so no additional imports are needed
  - Extend the `Thread` class (which implements the `Runnable` interface), and start the thread
    - `Thread` is in `java.lang`, so no additional imports are needed
- In either of the above cases
  - Call the `start()` method on the `Thread` class to kick off the new thread
  - Override the `public void run()` method in your new class
    - The `run()` method will be called when `start()` is called on your thread
    - What happens if you just call `run()` yourself instead of calling `start()`?
      - Your code will execute as if you just called a method, not using multi-threading

Thread Class



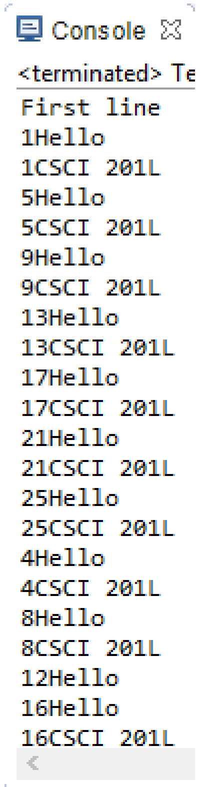
«interface»
java.lang.Runnable

java.lang.Thread

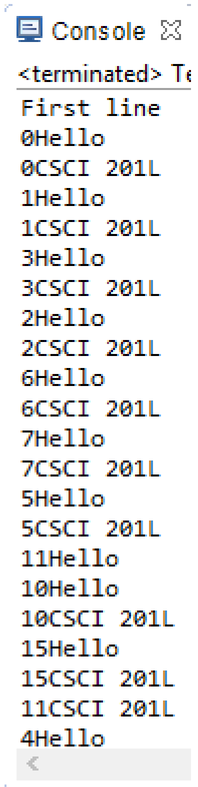
`+Thread()`	Creates an empty thread.
`+Thread(task: Runnable)`	Creates a thread for a specified task.
`+start(): void`	Starts the thread that causes the `run()` method to be invoked by the JVM.
`+isAlive(): boolean`	Tests whether the thread is currently running.
`+setPriority(p: int): void`	Sets priority `p` (ranging from 1 to 10) for this thread.
`+join(): void`	Waits for this thread to finish.
`+sleep(millis: long): void`	Puts a thread to sleep for a specified time in milliseconds.
`+yield(): void`	Causes a thread to pause temporarily and allow other threads to execute.
`+interrupt(): void`	Interrupts this thread.

Multi-Threaded Example - Thread

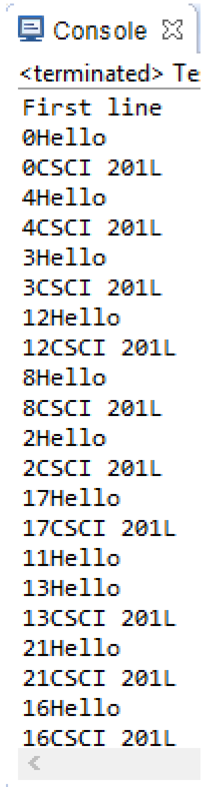
```java
public class Test {
  public static void main(String[] args ) {
    System.out.println("First line");
    for (int i=0; i < 100; i++) {
      Thread t = new TestThread(i);
      t.start();
    }
    System.out.println("Last line");
 }
}

class TestThread extends Thread {
  private int num;
  public TestThread(int num) {
    this.num = num;
  }
  public void run() {
    System.out.println(num + "Hello");
    System.out.println(num + "CSCI 201L");
  }
}
```



```
First line
1Hello
1CSCI 201L
5Hello
5CSCI 201L
9Hello
9CSCI 201L
13Hello
13CSCI 201L
17Hello
17CSCI 201L
21Hello
21CSCI 201L
25Hello
25CSCI 201L
4Hello
4CSCI 201L
8Hello
8CSCI 201L
12Hello
16Hello
16CSCI 201L
```



```
First line
0Hello
0CSCI 201L
1Hello
1CSCI 201L
3Hello
3CSCI 201L
2Hello
2CSCI 201L
6Hello
6CSCI 201L
7Hello
7CSCI 201L
5Hello
5CSCI 201L
11Hello
10Hello
10CSCI 201L
15Hello
15CSCI 201L
11CSCI 201L
4Hello
```



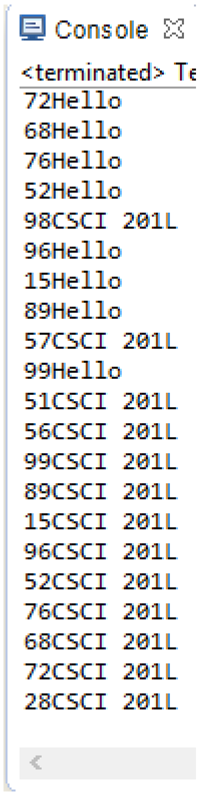
```
First line
0Hello
0CSCI 201L
4Hello
4CSCI 201L
3Hello
3CSCI 201L
12Hello
12CSCI 201L
8Hello
8CSCI 201L
2Hello
2CSCI 201L
17Hello
17CSCI 201L
11Hello
13Hello
13CSCI 201L
21Hello
21CSCI 201L
16Hello
16CSCI 201L
```

These are only the first 22 lines of output

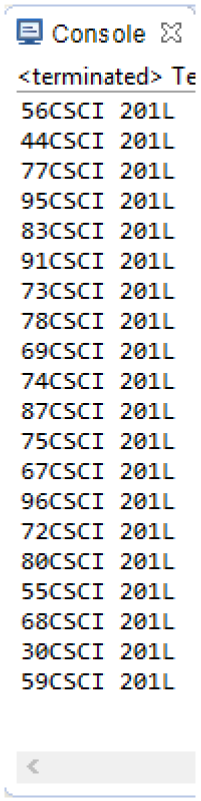
USC Viterbi
School of Engineering

Multi-Threaded Example - Runnable

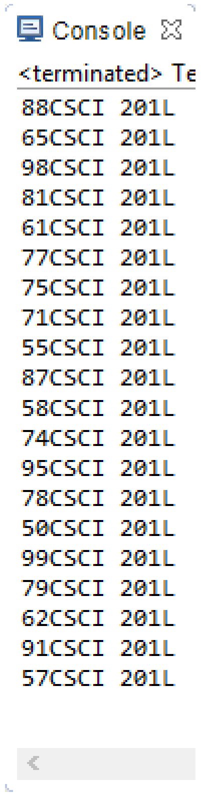
```java
public class Test {
  public static void main(String[] args ) {
    System.out.println("First line");
    for (int i=0; i < 100; i++) {
      Runnable r = new TestThread(i);
      Thread t = new Thread(r);
      t.start();
    }
    System.out.println("Last line");
  }
}

class TestThread implements Runnable {
  private int num;
  public TestThread(int num) {
    this.num = num;
  }
  public void run() {
    System.out.println(num + "Hello");
    System.out.println(num + "CSCI 201L");
  }
}
```



```
Console
<terminated> Te
72Hello
68Hello
76Hello
52Hello
98CSCI 201L
96Hello
15Hello
89Hello
57CSCI 201L
99Hello
51CSCI 201L
56CSCI 201L
99CSCI 201L
89CSCI 201L
15CSCI 201L
96CSCI 201L
52CSCI 201L
76CSCI 201L
68CSCI 201L
72CSCI 201L
28CSCI 201L
```



```
Console
<terminated> Te
56CSCI 201L
44CSCI 201L
77CSCI 201L
95CSCI 201L
83CSCI 201L
91CSCI 201L
73CSCI 201L
78CSCI 201L
69CSCI 201L
74CSCI 201L
87CSCI 201L
75CSCI 201L
67CSCI 201L
96CSCI 201L
72CSCI 201L
80CSCI 201L
55CSCI 201L
68CSCI 201L
30CSCI 201L
59CSCI 201L
```



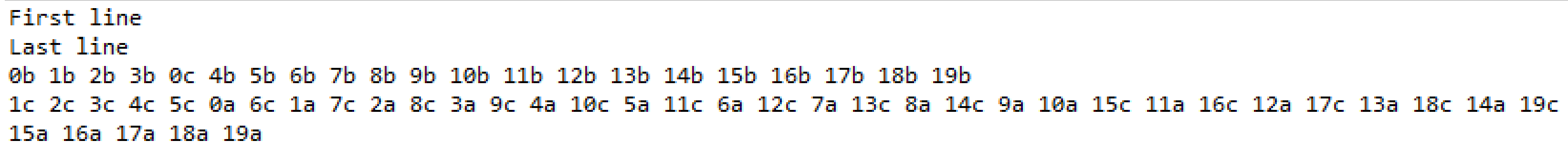
```
Console
<terminated> Te
88CSCI 201L
65CSCI 201L
98CSCI 201L
81CSCI 201L
61CSCI 201L
77CSCI 201L
75CSCI 201L
71CSCI 201L
55CSCI 201L
87CSCI 201L
58CSCI 201L
74CSCI 201L
95CSCI 201L
78CSCI 201L
50CSCI 201L
99CSCI 201L
79CSCI 201L
62CSCI 201L
91CSCI 201L
57CSCI 201L
```

These are only the last 20-21 lines of output

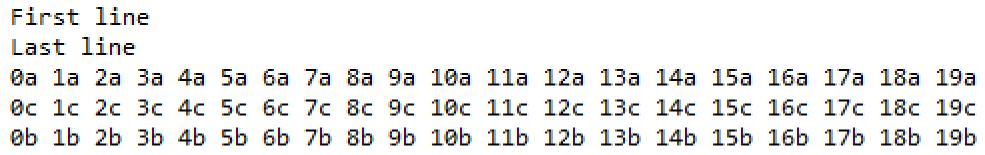
Multi-Threaded Example #2

```java
public class Test {
  public static void main(String[] args ) {
    System.out.println("First line");
    TestThread ta = new TestThread('a');
    TestThread tb = new TestThread('b');
    TestThread tc = new TestThread('c');
    ta.start();
    tb.start();
    tc.start();
    System.out.println("Last line");
  }
}
```

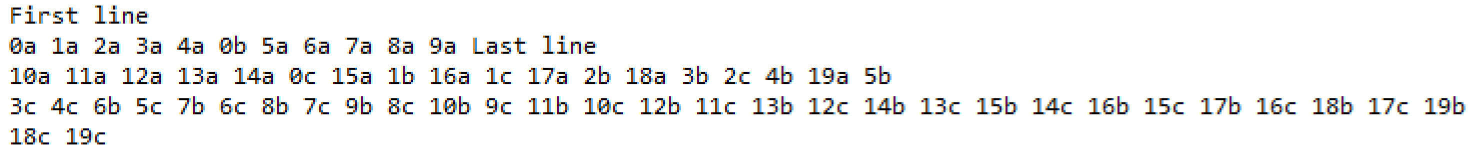
```java
class TestThread extends Thread {
  private char c;
  public TestThread(char c) {
    this.c = c;
  }
  public void run() {
    for (int i=0; i < 20; i++) {
      System.out.print(i + "" + c + " ");
    }
    System.out.println("");
  }
}
```



```
First line
Last line
0b 1b 2b 3b 0c 4b 5b 6b 7b 8b 9b 10b 11b 12b 13b 14b 15b 16b 17b 18b 19b
1c 2c 3c 4c 5c 0a 6c 1a 7c 2a 8c 3a 9c 4a 10c 5a 11c 6a 12c 7a 13c 8a 14c 9a 10a 15c 11a 16c 12a 17c 13a 18c 14a 19c
15a 16a 17a 18a 19a
```



```
First line
Last line
0a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12a 13a 14a 15a 16a 17a 18a 19a
0c 1c 2c 3c 4c 5c 6c 7c 8c 9c 10c 11c 12c 13c 14c 15c 16c 17c 18c 19c
0b 1b 2b 3b 4b 5b 6b 7b 8b 9b 10b 11b 12b 13b 14b 15b 16b 17b 18b 19b
```

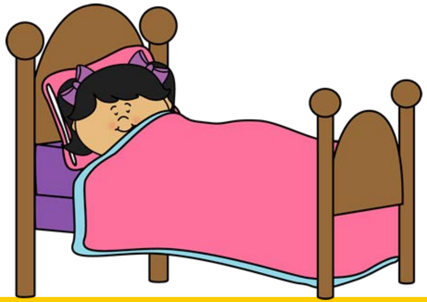


```
First line
0a 1a 2a 3a 4a 0b 5a 6a 7a 8a 9a Last line
10a 11a 12a 13a 14a 0c 15a 1b 16a 1c 17a 2b 18a 3b 2c 4b 19a 5b
3c 4c 6b 5c 7b 6c 8b 7c 9b 8c 10b 9c 11b 10c 12b 11c 13b 12c 14b 13c 15b 14c 16b 15c 17b 16c 18b 17c 19b
18c 19c
```

USC Viterbi
School of Engineering

`Thread.sleep(long)`

- The `sleep(long milliseconds)` method in the `Thread` class puts the thread to sleep for a number of milliseconds
- The `sleep` method does not cause the thread to be immediately switched back into the CPU after the number of milliseconds
  - From the sleeping state, the thread will go back into the ready state
  - This means that the thread will sleep for *at least* that many milliseconds
- The `sleep` method can throw an `InterruptedException` when another thread calls this thread's `interrupt()` method
  - The `interrupt()` method would get called when another thread wants that thread to stop whatever it is doing immediately

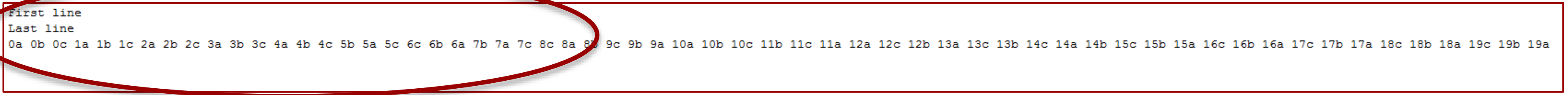


USC Viterbi School of Engineering

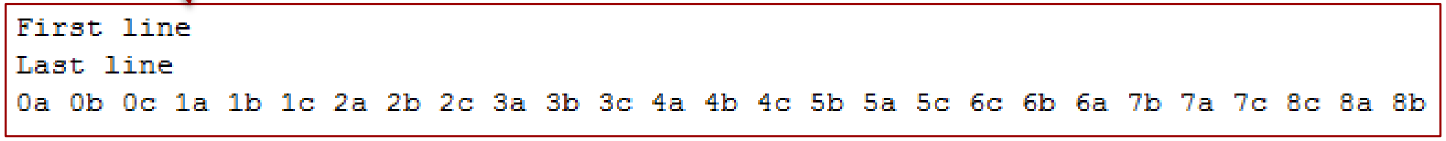
sleep() Example

```java
public class Test {
  public static void main(String[] args ) {
    System.out.println("First line");
    TestThread ta = new TestThread('a');
    TestThread tb = new TestThread('b');
    TestThread tc = new TestThread('c');
    ta.start();
    tb.start();
    tc.start();
    System.out.println("Last line");
  }
}
```

```java
class TestThread extends Thread {
  private char c;
  public TestThread(char c) {
    this.c = c;
  }
  public void run() {
    for (int i=0; i < 20; i++) {
      System.out.print(i + "" + c + " ");
      try {
        Thread.sleep(1000);
      } catch (InterruptedException ie) {
        System.out.println("interrupted");
        return;
      }
    System.out.println("");
  }
}
```



```
First line
Last line
0a 0b 0c 1a 1b 1c 2a 2b 2c 3a 3b 3c 4a 4b 4c 5b 5a 5c 6c 6b 6a 7b 7a 7c 8c 8a 8b 9c 9b 9a 10a 10b 10c 11b 11c 11a 12a 12c 12b 13a 13c 13b 14c 14a 14b 15c 15b 15a 16c 16b 16a 17c 17b 17a 18c 18b 18a 19c 19b 19a
```



```
First line
Last line
0a 0b 0c 1a 1b 1c 2a 2b 2c 3a 3b 3c 4a 4b 4c 5b 5a 5c 6c 6b 6a 7b 7a 7c 8c 8a 8b
```

Outline

- Threads
- Multi-Threaded Code
- CPU Scheduling
- Program

CPU Scheduling Overview

- The operating system will schedule different processes to execute in the CPU at different times
- Each process will execute for a given amount of time (or less if the entire time is not needed), then be switched out of the CPU so other processes don't starve
- The algorithm by which an operating system performs context switches is specific to the OS, though it can include **priority-based**, **round robin**, **fixed time**, **shortest remaining time**, **FIFO**, and many others
- It is important to note that an application programmer cannot determine when a process will be switched into the CPU but can only notify the operating system that the program is ready to be executed

Outline

- Threads
- Multi-Threaded Code
- CPU Scheduling
- Program

Program

- Write a program that prints out a line every 0.5 seconds. The lines could be hard-coded or read from a file. Print a timestamp with each line as well.

```
Hello CSCI 201.  The time in milliseconds is 1505823026561
Hello CSCI 201.  The time in milliseconds is 1505823027062
Hello CSCI 201.  The time in milliseconds is 1505823027562
Hello CSCI 201.  The time in milliseconds is 1505823028063
Hello CSCI 201.  The time in milliseconds is 1505823028563
Hello CSCI 201.  The time in milliseconds is 1505823029063
Hello CSCI 201.  The time in milliseconds is 1505823029563
Hello CSCI 201.  The time in milliseconds is 1505823030064
Hello CSCI 201.  The time in milliseconds is 1505823030565
Hello CSCI 201.  The time in milliseconds is 1505823031065
```