Lec 17

Threads
Single processor

CPU

I/O

Memory
Multi processes

Eclipse

PPT

iClicker
Multi processor
Multi-core
Logical Cores

Task Manager

- CPU: 6% 0.78 GHz
- Memory: 3.4/3.9 GB (87%)
- Disk 0 (C): 0%
- Wi-Fi: S: 0 R: 8.0 Kbps
- Ethernet: Not connected
- Bluetooth: Not connected
- Ethernet: S: 0 R: 0 Kbps

CPU: Intel(R) Core(TM) i3-4010U CPU @ 1.70GHz

Utilization: 6%
Speed: 0.78 GHz
Processes: 116
Threads: 1198
Handles: 73840

Up time: 4:19:45:51

Maximum speed: 1.70 GHz
Sockets: 1
Cores: 2
Logical processors: 4
Virtualization: Disabled
Hyper-V support: Yes

L1 cache: 128 KB
L2 cache: 512 KB
L3 cache: 3.0 MB
Multi-threaded
Creating Threads

1. Write code for thread to execute

   public class Example implements Runnable {
       public void run() {
           // your code here
       }
   }

2. Create the thread and link to (1)

   Example ex = new Example();
   Thread thread = new Thread(ex);

3. Start the thread

   thread.start();
public class PrintChar implements Runnable {
    private char charToPrint;

    public PrintChar(char c) {
        charToPrint = c;
    }

    public void run() {
        for(int i = 0; i < 5; i++) {
            System.out.print(charToPrint);
        }
    }
}

public class Example {
    public static void main(String[] args) {
        PrintChar a = new PrintChar('a');
        PrintChar b = new PrintChar('b');

        Thread t1 = new Thread(a);
        Thread t2 = new Thread(b);
        t2.start();
        t1.start();
    }
}
main()
main()
t2.start()
main()
t2.start();
t1.start()
Creating Theads, pt II

• Create a class that extends Thread and implements run()

```java
public class Example extends Thread {
    public void run() {
        // your code here
    }
}
```

• Start the thread

```java
Thread thread = new Example();
thread.start();
```
public class Sum extends Thread {
    int start, stop, sum;
    public Sum(int start, int stop) {
        // initialize vars.
    }
    public void run() {
        for(int i = start; i <= stop; i++) {
            sum += i;
        }
    }
}

public class Example {
    private static final int SUM_NUMBERS = 5;
    public static void main(String[] args) {
        Thread t1 = new Sum(0, SUM_NUMBERS);
        t1.start();
        System.out.println(((Sum) t1).sum); // note: this is ok!
    }
}

What gets printed

A) Compiler error
B) 0
C) 15
D) Number between 0 and 15
E) B, C, or D
public class Example extends Thread {
    public Example() {
        //initialize class.
    }
    public void run() {
        try {
            sleep(10); //make the thread sleep for 10ms
        }
        catch(InterruptedException e) {
        }
    }
}
Thread thread = new Sum(0, SUM_NUMBERS);
thread.start();

try {
    thread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}