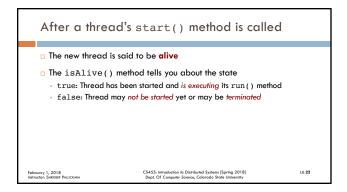


Starting a thread (2/2)

When we're ready for a thread to begin executing code
Call the start() method
start() performs internal house-keeping and then calls the run() method
When the start() method returns?
Two threads are executing in parallel
1 The original thread which just returned from calling start()
The newly started thread that is executing its run() method

| Starting | 1,018 | C4455. Introduction to Distributed Systems [Spring 2018] | Days. Of Computer Science, Colorado Stare University



Terminating a thread

Once started, a thread executes only one method: run()

This run() may be complicated

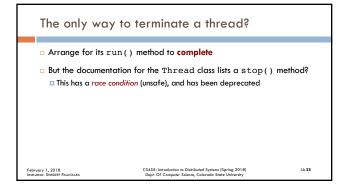
May execute forever

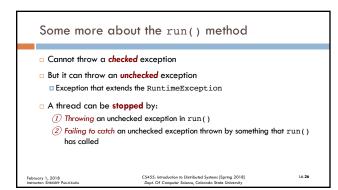
Call several other methods

Once the run() finishes executing, the thread has completed its execution

Like all Java methods, run() finishes when it ...

1 Executes a return statement
2 Executes the last statement in its method body
3 When it throws an exception
Or fails to catch an exception thrown to it





Pausing, suspending and resuming threads

Some thread models support the concept of thread suspension
Thread is told to pause execution and then told to resume its execution
Thread contains suspend() and resume()
Suffers from vulnerability to race conditions: deprecated
Thread can suspend its own execution for a specified period
By calling the sleep() method

Patronary 1, 2018
Dayl Of Compute Source, Colorado State University

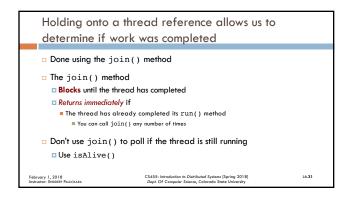
But sleeping is not the same thing as thread suspension

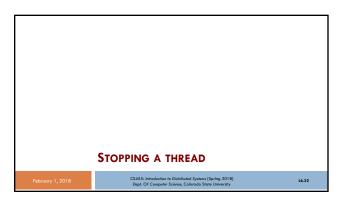
With true thread suspension
One thread can suspend (and later resume) another thread
sleep() affects only the thread that executes it
Not possible to tell another thread to go to sleep

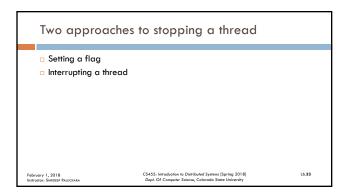
But you can achieve the functionality of suspension and resumption

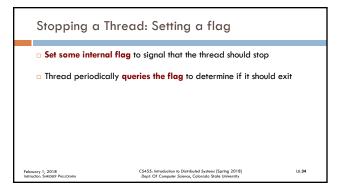
Use wait and notify mechanisms

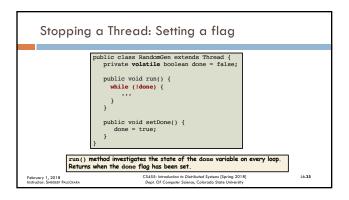
Threads must be coded to use this technique
This is not a generic suspend/resume that is imposed by another thread

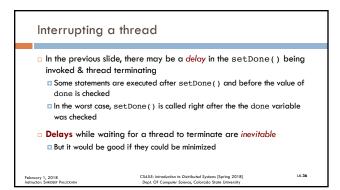


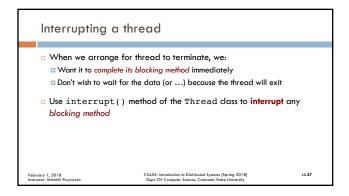


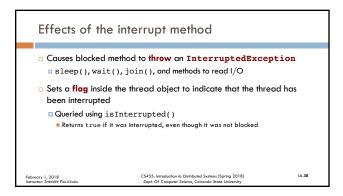


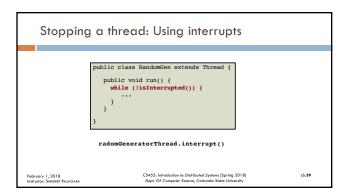


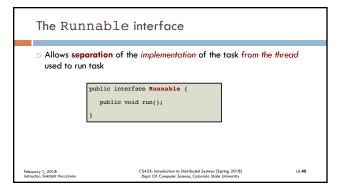


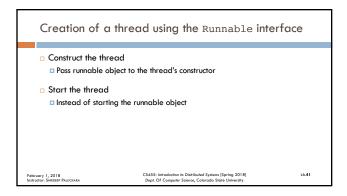








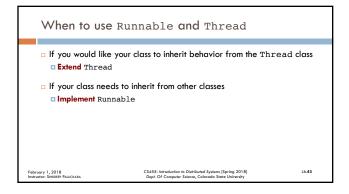


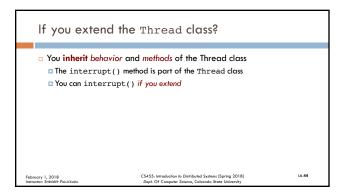


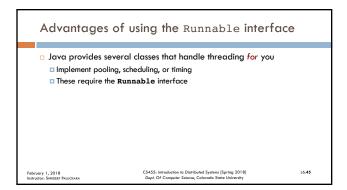
```
Creation of a thread using the Runnable interface

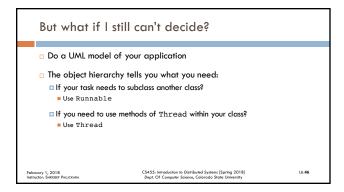
public class RandomGenerator implements Runnable {
    public void run() { ... }
}
...
generator = new RandomGenerator();
Thread createdThread = new Thread(generator);
createdThread.start();

February 1, 2018
Located States (State State State
```





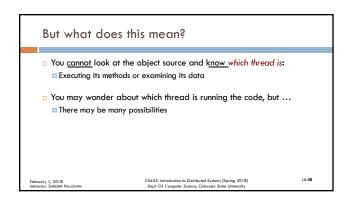


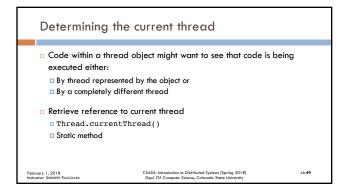


Threads and Objects

Instance of the Thread class is just an object
Can be passed to other methods
If a thread has a reference to another thread
In the can invoke any method of that thread's object
The Thread object is not the thread itself
In the set of methods and data that encapsulate information about the thread

February 1, 2018
Days of Canapher Sissens, Calenda Systems (Spring 2018)
Days of Canapher Sissens, Calenda State University





```
Allowing a Runnable object to see if it has been interrupted

public class MyRunnable implements Runnable {
    public void run() {
        if (!Thread.currentThread().isInterrupted() ) {
            ... Main logic
        }
    }
}

February 1, 2018
C5455: Introduction to Distributed Systems (Spring 2018)
Days Of Computer Sistems, Colorado Status University
```

```
The contents of this slide-set are based on the following references

Java Threads. Scott Oaks and Henry Wong. . 3rd Edition. O'Reilly Press. ISBN: 0-596-00782-5/978-0-596-00782-9. [Chapters 3, 4]
Operating Systems Principles and Practice. Thomas Anderson and Michael Dahlin. 2nd Edition. ISBN: 978-0985673529. [Chapter 4]
```