



3C03 Concurrency: Concurrency in Java

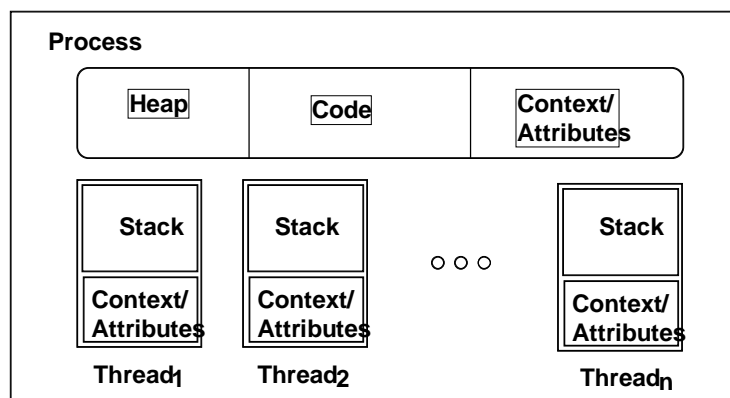
Wolfgang Emmerich

© Wolfgang Emmerich, 1998/99

1



Threads and OS Processes



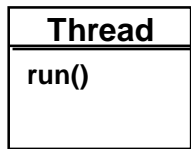
- OS process provides protected address space.
- Many threads may execute within space.
- Each thread: stack & context (saved registers).

© Wolfgang Emmerich, 1998/99

2



Threads using Inheritance



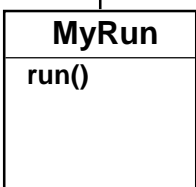
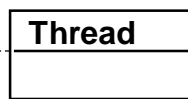
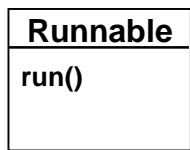
```
class MyThread extends Thread {
    public void run() {
        ...
    }
}
```

Creation of thread:

```
MyThread t=new MyThread();
```



Threads implementing Interfaces



```
class MyRun implements Runnable {
    public void run() {
        ...
    }
}
```

Creation of thread:

```
Thread t=new Thread(new MyRun);
```



Thread Lifecycle

- **Started by `start()` which invokes `run()`**
- **Terminated when**
 - `run()` returns
- **A started thread may be**
 - `running` or
 - `runnable` (waiting to be scheduled)
- **Thread gives up processor using `yield()`.**
- **`sleep()` suspends for a given time and then resumes**
- **`sleep()` without argument suspends**
- **`wake()` resumes the thread**

© Wolfgang Emmerich, 1998/99

5



FSP Model of Java Thread Lifecycle

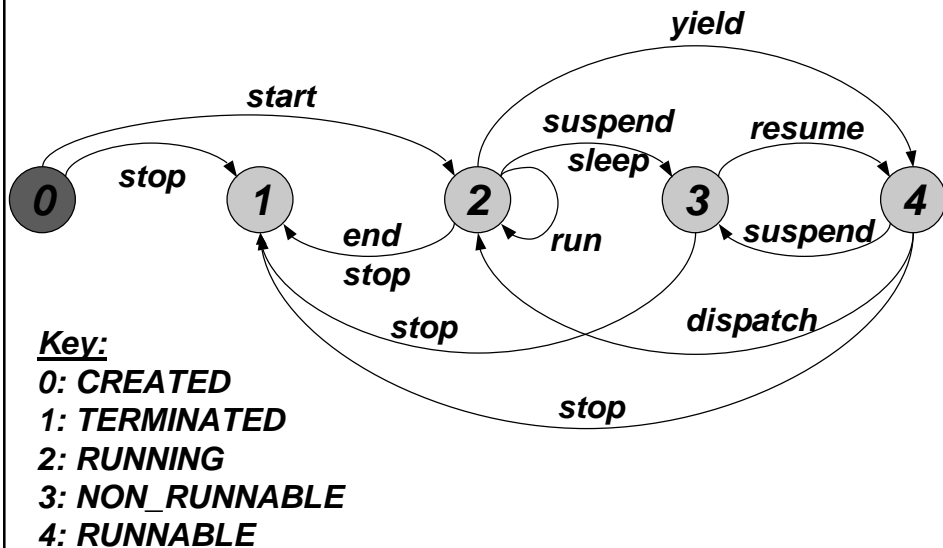
```
THREAD = CREATED,  
CREATED = ( start -> RUNNING  
            | stop -> TERMINATED ),  
RUNNING = ( suspend -> NON_RUNNABLE  
            | sleep -> NON_RUNNABLE  
            | yield -> RUNNABLE  
            | {stop, end} -> TERMINATED  
            | run -> RUNNING ),  
RUNNABLE = ( suspend -> NON_RUNNABLE  
            | dispatch -> RUNNING  
            | stop -> TERMINATED ),  
NON_RUNNABLE = ( resume -> RUNNABLE  
                | stop -> TERMINATED ),  
TERMINATED = STOP.
```

© Wolfgang Emmerich, 1998/99

6



LTS of Java Thread Lifecycle



Key:

- 0: CREATED
- 1: TERMINATED
- 2: RUNNING
- 3: NON_RUNNABLE
- 4: RUNNABLE

© Wolfgang Emmerich, 1998/99

7



Example: Countdown Timer

■ **Demo: Countdown**

■ **FSP of Countdown:**

```

COUNTDOWN (N=10) = COUNTDOWN[N],
COUNTDOWN[i:0..N] =
  ( when(i>0) tick->COUNTDOWN[i-1]
  | when(i==0) beep->STOP
  ).

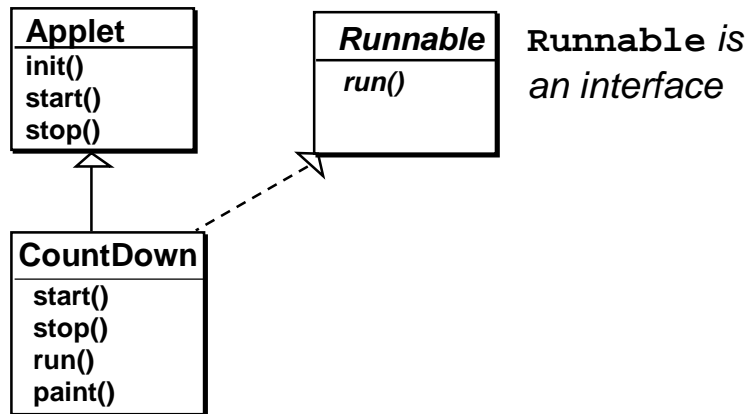
```

© Wolfgang Emmerich, 1998/99

8



CountDown Timer - Class diagram



© Wolfgang Emmerich, 1998/99

9



CountDown Timer - Java class

```
import java.awt.*;           //windows toolkit
import java.applet.*;       //applet support
public class Countdown extends Applet implements Runnable{
    int counter; Thread cd;
    public void start() {      // create thread
        counter = 60; cd = new Thread(this); cd.start();
    }
    public void stop() { cd = null;}
    public void run() {       // executed by Thread
        while (counter>0 && cd!=null) {
            try{Thread.sleep(1000);}
            catch (InterruptedException e){}
            --counter; repaint(); //update screen
        }
    }
    public void paint(Graphics g) {
        if (counter>0)
            g.drawString(String.valueOf(counter),25,75);
        else g.drawString("Bang", 10, 50);
    }
}
```

© Wolfgang Emmerich, 1998/99

10



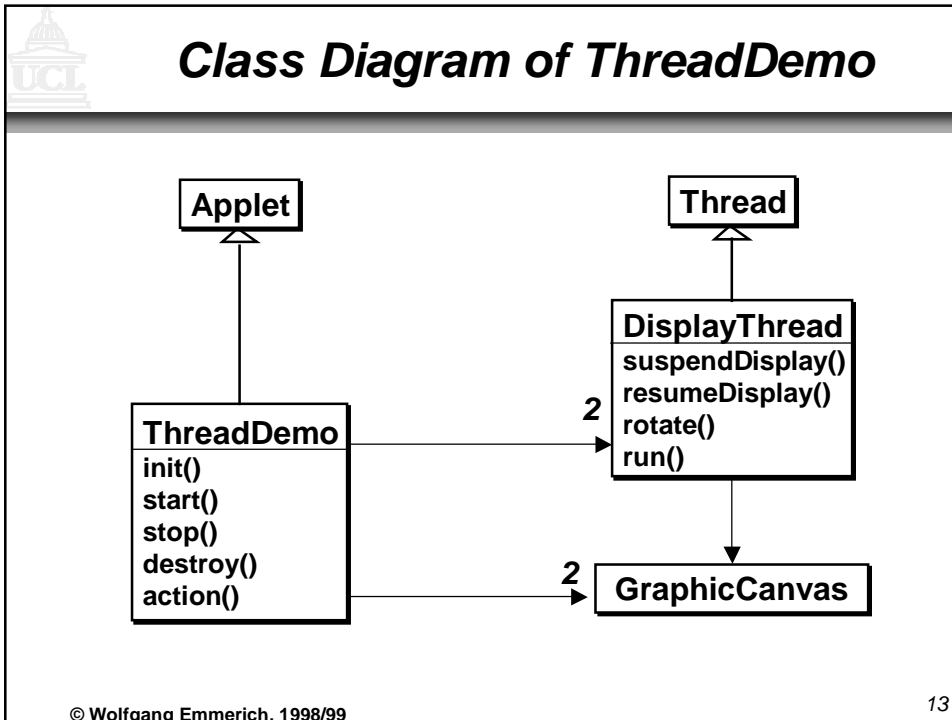
Concurrent Threads


- *Parallel composition operator* | |
- *Implemented by creation of several new thread objects*
- **Example: ThreadDemo**
- *Creates two thread objects that execute concurrently*



FSP Spec of Thread Demo

```
DISPLAY_THREAD = SUSPENDED,  
SUSPENDED = ( resume->RUNNING ),  
RUNNING = ( rotate->RUNNING  
           | suspend->SUSPENDED  
           ).  
| | THREAD_DEMO =  
    ( a:DISPLAY_THREAD | | b:DISPLAY_THREAD ).
```



- Summary**
- 
- ***Threads vs. operating system processes***
 - ***Threads through class inheritance / interface implementation***
 - ***Thread lifecycle***
 - ***Concurrent threads by creating new thread objects***
 - ***Class diagrams***
 - ***Next: Java Thread Programming Lab***
- © Wolfgang Emmerich, 1998/99 14