3C03 Concurrency: Concurrency in Java

Mark Handley

Threads and OS Processes

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

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

Creation of thread:
MyThread t = new MyThread();
```

Threads implementing Interfaces

```java
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
  - 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.

FSP Model of Java Thread Lifecycle

```
THREAD = CREATED,
CREATED = ( start -> RUNNING
            | stop -> TERMINATED),
RUNNING = ( sleep -> NON_RUNNABLE
           | yield -> RUNNABLE
           |{stop, end} -> TERMINATED
           | run -> RUNNING),
RUNNABLE= ( sleep -> NON_RUNNABLE
           | dispatch -> RUNNING
           | stop -> TERMINATED),
NON_RUNNABLE = ( wake ->RUNNABLE
                | stop -> TERMINATED),
TERMINATED = STOP.
```
LTS of Java Thread Lifecycle

Example: CountDown Timer

**Demo: CountDown**

FSP of CountDown:

\[
\text{COUNTDOWN (N=10) = COUNTDOWN[N],}
\]
\[
\text{COUNTDOWN[i:0..N] =}
\]
\[
\begin{align*}
(\text{ when}(i>0) \text{ tick}\rightarrow \text{COUNTDOWN[i-1]} \\
| \text{ when}(i==0) \text{ beep}\rightarrow \text{STOP} ) .
\end{align*}
\]
CountDown Timer - Class diagram

Applet
- init()
- start()
- stop()

Runnable
- run()

CountDown
- start()
- step()
- run()
- paint()

Runnable is an interface

CountDown Timer - Java class

```java
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, Mark Handley 1998 - 2003
Concurrent Threads

- Parallel composition operator \( || \)
- Implemented by creation of several new thread objects

- Creates two thread objects that execute concurrently
  Example: ThreadDemo

FSP Specification of Thread Demo

\[
\text{DISPLAY_THREAD = SUSPENDED,} \\
\text{SUSPENDED = ( resume->RUNNING ),} \\
\text{RUNNING = ( rotate->RUNNING} \\
\text{ | suspend->SUSPENDED)} \\
\text{ | | THREAD_DEMO =} \\
\text{(a:DISPLAY_THREAD | b:DISPLAY_THREAD).}
\]
Class Diagram of ThreadDemo

- Applet
  - ThreadDemo
    - init()
    - start()
    - stop()
    - action()
  - DisplayThread
    - suspendDisplay()
    - resumeDisplay()
    - rotate()
    - run()
- Thread
  - GraphicCanvas

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