CS1007: Object Oriented Design and Programming in Java

Lecture #22

Apr 11

Shlomo Hershkop
shlomo@cs.columbia.edu

Outline

- Multi threaded models:
- Sharing resources
- Sending signal
- Threaded model
- Java code
- More java code

- Reading: Chapter 9-9.2

Feedback

- Thursday will be small class on graphic programming tips and open ended programming help from TA's
  - Will make sure you are at least started on the assignment
  - Will have some code for you to use if you need
  - Double points if you stump them ☺
  - Also: please submit via email (to me) a screenshot of your Othello game GUI, we will have a vote in the last class for best of show.

Homework

- How are you testing for next move ?

Homework

- How are you saving the game ?

Threading

- Program which run on your system are scheduled in such a way that they you are given the impression that it is always being executed
- In truth there is a system called the scheduler which decides which program gets to run
- No guarantees

- We discussed some simple scheduling algorithms last class
- So the resource gets shared between programs running at the same time
- Is this always good?

- Works for something like CPU
- What If the resource is a printer ??

- So assume your cpu is being scheduled between programs, how can one program tell the scheduler it is done
- Or allow the scheduler to reschedule someone's slot in middle of running

Signaling

- Need system to allow processes to signal system when done, or problems
- Anyone hear of interrupts ?? IRQ

Process

- Your running program
- Has memory space associated with it
- Variables
- State
  - Processor state
  - Memory state
- Set of resources
- Permission level

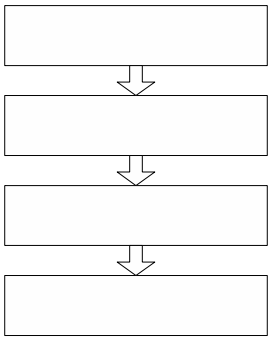
Threads vs processes

- If you want to run multiple process expensive to switch context
- Java allows you to launch multiple threads of execution
  - Processes don't share memory space
  - Really all the threads are under java's execution process (some exceptions)

What is a thread

- So what exactly is a thread?
- Think of it as mini program within your program
- Exciting thing: can run multiple mini programs inside your program at the same time

Sequential program



Sequential instructions

- Pro:
  - Easier to read
  - When things go wrong, easier to pinpoint error
  - easier to debug
- Con:
  - May have long waits at points
    - I/O bound problem
  - Something important might get lost
  - Not as cool ☺
  - Might break your computer

Threaded instructions

- Can get more done at the same time
- More efficient
- Harder to debug, as specific condition (race) might be hard to replicate

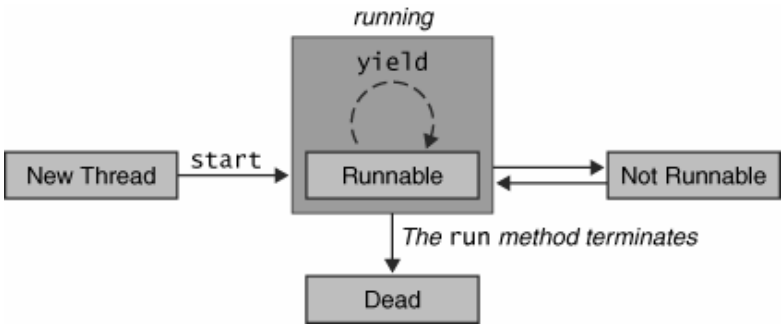
- Chapter 9 of book ☺

Life of a thread

- Not what you think!

Stages

1. New
2. Runnable
3. Blocked
4. Dead



New stage

- So create the thread using some code
- Will do nothing so far
  - Maybe crash

runnable

- Your thread is running
- Start with call to `start()` method

Not runnable

- Going to sleep ☺
- Wait for something specific to happen
  - Getting a lock
  - Waiting for some other process to get results
  - etc
- Blocking and waiting for I/O

Going back to running

- Once a thread isnt running
- Scheduler will choose a thread

- Based on priority
- Based on schedule

Dead stage

- Run method terminates
  - normally

Idea

- Have some set of java instructions you want to run
- Put it in a thread
- Start the thread

- Example: Number X….is it prime
- Algorithm 1 - will find answer in 100 seconds
- Algorithm 2 - works between 50 - 500 seconds
  - Instead of having to choose one, just run both at the same time ☺

Nice thread

- A nice thread will make place for others to run by not hogging CPU
- Implemented on Linux OS

- Called yield in java
  - Will only apply if same or higher priority threads waiting to run

sleeping

- Not during class ☺
- Will allow itself to be put in the background so other threads can run
- Thread.sleep(milliseconds)
  - Will sleep for X milliseconds
  - Can be interrupts with exception/signal
  - Need to think about how to handle that

Coding

- 2 ways to code threads in java
- Extend Thread
  - Override run method
- Implement runnable
- We will now walk through both

Problem

- Cant decide what you want to major in
- Will run 2 threads, which ever finishes first will choose
- What do you think ?

```
public class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }

    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + getName());
            try {
                sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! “ + getName());
    }
}
```

```
public class TwoThreadsTest {
    public static void main (String[] args) {
    new SimpleThread(“Computer Science").start();
    new SimpleThread(“Engineering").start();
    }
}
```

- Other option to implement runnable
- Provide a class that implements the Runnable interface and therefore implements the run method
- In this case, a Runnable object provides the run method to the thread

- Code example

Testing state

- Thread.getState()

- NEW
- RUNNABLE
- BLOCKED
- WAITING
- TIMED_WAITING
- TERMINATED

isAlive()

- True
  - Runnable
  - not runnable
- False
  - New state
  - terminated

Priority

- In addition to state, each thread has a priority associated with it
- Can change the threads priority manually
- No guarantee on anything
- Range:
  - MIN_PRIORITY
  - MAX_PRIORITY

Question

- Initially Thread class had stop() method to stop a running thread
- It has been removed…any ideas why ?
- So how can we stop a running thread ?

Signaling threads

- X.interrupt();
- Calling sleep will trigger an interrup exception
- Can manually look it up
  - Thread.currentThread().inInterrupted()

Warning

- Don’t blindly ignore interrupts
- Deal with them
- Can also set interrupts after catching if want to deal with it elsewhere in your code

Race condition

- One more new thing to worry about (unlike sequential instructions)
- When 2 threads simultaneously try to change a single object, leaving the resulting state undefined
- Example

- Any ideas of what to do ?

Airline bathroom

- Imagine the resource is an airline bathroom
- Only one thread at a time
- When want to use it:
  - If free …grab
  - Else wait (FIFO)

Locks

- Need a system for locking down a resource
- Don’t want anyone writing to the file while you are reading it
- Avoid inconsistency!!

ReentrantLock()

- Objects you create
- Call lock at the beginning of a block
- Make sure to call unlock at end
- Or use the finally after a catch block

- Deadlock problem
- Can go to sleep right after a check
- When come back wont be true anymore

Synchronized keyword

- Allow you to create a mutually excusive lock on a block of code
- Any part of the program which want to enter this ‘zone’ needs to aquire the lock
- Else wait till its free
- Some overhead
  - Leave out of loop code

Thursday

- Please start working on the homework
- TAs will present some graphic programming tips and help you with the homework