Java - Threads

CS 4354
Fall 2012
Jill Seaman

Threads

• What is a thread?
  ✦ A thread is an execution stream within a process.

• A thread is also called a lightweight process.
  ✦ Has its own execution stack, local variables, and program counter.
  ✦ Very much like a process, but it runs within a process.

Multithreading

• There may be more than one thread in a process.
  ✦ Is called a multithreaded process.
  ✦ Multithreading provides the capability to run tasks in parallel for a process.
  ✦ All threads of a process share with each other resources allocated to the process.
  ✦ In fact, they compete with each other.

• Threads allow the programmer to turn a program into separate, independently running subtasks

Threads in Java

• java.lang.Thread has all the wiring necessary to create and run threads.

• The run() method contains the code that will be executed “simultaneously” with the other threads in a program

• The Java Thread class implements a generic thread that, by default, does nothing.
  ✦ Its run() method is empty.
Threads in Java

- There are two techniques to implement threads in Java:
  ✦ To subclass Thread and override run().
  ✦ To implement the Runnable interface (by defining run()) and embed class instances in a Thread object.

- Once a Thread instance is created, call the start() method to make it run.
  ✦ This causes the run() method to be executed in a separate thread.
  ✦ The code following the call to start() will execute concurrently with the thread’s run method.

---

Subclassing Thread: example

```java
public class YinYang extends Thread {
    private String word; // what to say
    public YinYang(String whatToSay) {
        word = whatToSay;
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.print(word + " ");
            yield(); // to give another thread a chance
        }
    }
    public static void main(String[] args) {
        YinYang yin = new YinYang("Yin"); // to create Yin thread
        YinYang yang = new YinYang("Yang"); // to create Yang thread
        yin.start(); // to start Yin thread
        yang.start(); // to start Yang thread
    }
}
```

**output:**

```
Yin Yang Yang Yang Yin Yang Yin Yang Yin Yin
```

---

Implementing Runnable: example

```java
public class YangYin implements Runnable {
    private String word; // what to say
    public YangYin(String whatToSay) {
        word = whatToSay;
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.print(word + " ");
            Thread.yield(); // to give another thread a chance
        }
    }
    public static void main(String[] args) {
        Runnable rYang = new YangYin("Yang"); // to instantiate YangYin
        Runnable rYin = new YangYin("Yin"); // to instantiate again
        Thread yang = new Thread(rYang); // to create Yang thread
        Thread yin = new Thread(rYin); // to create Yin thread
        yang.start(); // to start Yang thread
        yin.start(); // to start Yin thread
    }
}
```

**output:**

```
Yin Yin Yang Yin Yang Yin Yang Yin Yin
```

---

Thread methods

- run()
  ✦ The code that will be run concurrently (in its own thread)
- start()
  ✦ Causes the run method to execute in its own thread, continues execution.
- yield()
  ✦ Causes the currently executing thread object to temporarily pause and allow other threads to execute.
- sleep(long millis)
  ✦ Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds
Thread methods

- **getName()**
  ✦ Returns this thread’s name.
  ✦ Can be set in the constructor (else it gets generated).
  ✦ Not necessarily unique.

- **interrupt()**
  ✦ Called from outside the thread.
  ✦ Interrupts a thread that is paused via sleep(), wait(), or join().
  ✦ InterruptedException is generated in the sleep/wait/join
  ✦ Calls to these methods must be in a try/catch block

Thread methods

- **join()**
  ✦ One thread may call join() on another thread to wait for the second thread to complete before proceeding.
  ✦ If a thread calls t.join() on another thread t, then the calling thread is suspended until the target thread t finishes

Thread synchronization

- We now have the possibility of two or more threads trying to use the same limited resource at once.
  ✦ i.e. two threads trying to access the same bank account at the same time

- If a certain method should not be called from two threads at the same time, you can use the keyword “synchronized”.

- If a thread is inside one of the synchronized methods, all other threads are blocked from entering any of the synchronized methods of the class until the first thread returns from its call

```java
public class CriticalCode {
    private int i;
    public synchronized void next() { i++; i++; }
    public synchronized int getValue() { return i; }
    ...
}
```