Inter-Thread Communication

ComS 587X
Fall, 2002

Pipes

- Java PipedOutputStream and PipedInputStream
  - Allow unidirectional messages between threads
  - Use like a regular InputStream or OutputStream
  - Bidirectional communication requires two pipes
- One end of pipe must be passed to the other thread somehow
  - E.g., via the constructor for the new Runnable object

Pipe Demo

```
public class PipeDemo extends Thread {
 PipedOutputStream output;
 public PipeDemo(PipedOutputStream out) { output = out; }
 public static void main (String args[]) {
  try { PipedOutputStream pout = new PipedOutputStream();
   PipedInputStream pin = new PipedInputStream(pout);
   PipeDemo pipedemo = new PipeDemo(pout);
   pipedemo.start();
   while ((int input = pin.read()) != -1) {
    System.out.print ( (char) input);
   } } catch (Exception e) {
    System.err.println ("Pipe error" + e);
   } }
 public void run() { try {
   PrintStream p = new PrintStream(output);
   p.println("Hello from the other thread, via pipes!");
   p.close(); } catch (Exception e) { /* Nothing */ } } }
```

Thread Waiting

- Most often, threads wait for events from another thread
  - E.g., for access to new data
- Simplest solution: Wait for termination of thread via join() method
  - Not very useful, though
- Better solution: wait() and notify()/notifyAll()
  - Queue of waiting threads maintained for each object

Wait and Notify

- First, must have lock on object
  - Via synchronized method or synchronized block
- Wait indefinitely:
  - object.wait();
- Wait for x milliseconds:
  - object.wait(x);
- Wake all waiting threads
  - object.notifyAll();
- Wake one waiting thread
  - object.notify();

Wait Example

```java
public class WaitNotify extends Thread {
 public static void main(String args[]) throws Exception{
  Thread notificationThread = new WaitNotify();
  notificationThread.start();
  synchronized (notificationThread) {
   notificationThread.wait(); }
  System.out.println ("The wait is over");
 }
 public void run() {
  System.out.println ("Hit enter to stop waiting thread");
  try { System.in.read(); }
  catch (java.io.IOException ioe) { /* No code */}
  synchronized (this) { this.notifyAll(); }
 }
}
```

Thread Groups

- Threads may be grouped to allow management of entire group
  - Rather than managing each thread individually
- ThreadGroup class
  - When new Threads are constructed, a reference to a ThreadGroup may be given to include the new thread in the group
- ThreadGroups may also contain ThreadGroups

ThreadGroup Methods

- Constructors
  - public ThreadGroup(String name)
  - public ThreadGroup(ThreadGroup parentGroup, String name)
- Methods
  - int activeCount() - count active threads in group and subgroups
  - int activeGroupCount() - count groups with active threads
  - boolean allowThreadSuspension() - is suspension allowed?
  - void checkAccess() - may the ThreadGroup be modified?
  - void destroy() - destroy ThreadGroup and its subgroups
  - int enumerate(Thread[] threadList) – get array of Threads
  - int enumerate(Thread[] threadList, boolean subGroupFlag)
  - int enumerate(ThreadGroup[] groupList)
  - int enumerate(ThreadGroup[] groupList, boolean subGroupFlag)

ThreadGroup Methods (2)

- int getMaxPriority() - max priority level of threads in the group
- String getName() - return the name of the ThreadGroup
- ThreadGroup getParent() - obtain the parent group
- void interrupt() - invoke the interrupt() method on all threads
- boolean isDaemon() - true if group is a daemon group
- boolean isDestroyed() - true if the group has been destroyed
- void list() - dump info about group to System.out
- boolean parentOf(ThreadGroup otherGroup) – test relationship
- void resume() - resume all threads
- void setDaemon(boolean flag) – set daemon mode on group
- void setMaxPriority(int priority) – limit max priority of any thread
- void stop() - stop all threads
- void suspend() - suspend all threads
- void uncaughtException(Thread t, Throwable error) – called when a thread fails to catch a runtime exception

ThreadGroup Example

```
public class GroupDemo implements Runnable {
 public static void main(String args[]) throws Exception{
  ThreadGroup p = new ThreadGroup("parent");
  ThreadGroup sg = new ThreadGroup(p, "subgroup”);
  Thread t1 = new Thread(p, new GroupDemo()); t1.start();
  Thread t2 = new Thread(p, new GroupDemo()); t2.start();
  Thread t3 = new Thread(sg, new GroupDemo()); t3.start();
  parent.list();
  System.out.println ("Press enter to continue");
  System.in.read();
  System.exit(0);
 }
public void run() {
  for( ; ; ) {
   Thread.yield();
  }
 }
}
```

Thread Priorities

- Priorities range from 10 (Thread.MAX_PRIORITY) through 5 (Thread.NORM_PRIORITY) to 1 (Thread.MIN_PRIORITY)
- Example: Raise current thread's priority

```
Thread t = Thread.currentThread();
t.setPriority(Thread.MAX_PRIORITY);
```

Summary

- Thread communication
  - Pipes
  - wait(), notify(), notifyAll()
- Thread groups
- Thread priorities