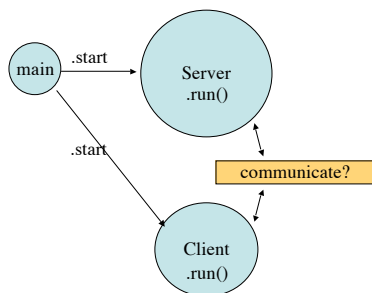


Threads
read Chapter 9
(sorry... not Chapter 10)

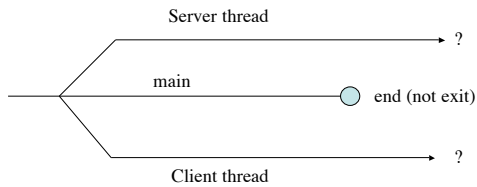
what are threads?

- independently running programs within one Unix process
- can communicate through static variables that they share
- define the “Client / Server” relationship
- used when a task needs undivided attention
 - playing mp3s
 - running a task while waiting for user input

how threads operate



timeline



writing a client thread

```
public class ClientClass extends Thread
{
    public void run()
    {
        System.out.println("Starting Client");
        do while
        {
            // stuff
        } (true);
    }
}
```

writing a Server

```
public class ServerClass extends Thread {
    public void run() {
        while {
            // accept commands here
        } (true);
    }
    public static void main (String [] args) {
        ClientClass clientThread = new ClientClass();
        clientThread.start();
        ServerClass serverThread = new ServerClass();
        serverThread.start();
    } // end main
} // end class
```

your Server Class responsibility

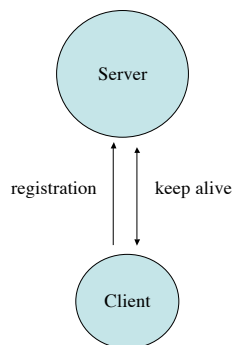
- main
- start own thread
- start all client threads
- run forever until commanded to stop
- System.exit(0) from this class's .run() ends all threads
 - actually any System.exit(0) ends all threads.
- many ways to command - shared variables are often used; files; etc.

talk to each other

```
public class commClass
{
public static String ClientToServer = "idle";
public static String ServerToClient = "idle";
} // end commClass
```

- static means everyone can use the Strings
 - commClass.ClientToServer = "exit";
- defines a "protocol"

typical



some useful methods

`Thread.sleep(milliseconds)` - suspends this thread's activity

stopping a thread

- a clean end to a threads `.run()` method ends that thread.
- `System.exit(0)` from ANYWHERE ends all threads
