1. Creating and running a thread
   Thread th = new Thread (new Runnable ()
   {
     public void run ()
     {
       ... action here ...
     }
   });

2. Stopping a thread (exit variable)
   private boolean m_isRunning;
   public synchronized void setIsRunning (boolean newVal)
   {
     m_isRunning = newVal;
   }
   public synchronized boolean IsRunning ()
   {
     return m_isRunning;
   }
   public void run ()
   {
     setIsRunning(true);
     while (isRunning())
     {
       ... do something ...
     }
   }
   To stop the thread:
   thread.setIsRunning (false);

3. Stopping a thread (interrupt)
   public void run ()
   {
     try
     {
       while (true)
       {
         foo.wait();
         ... do something ...
       }
     } catch (InterruptedException e)
     {
       .. clean up ...
     }
   }
   To stop the thread:
   thread.interrupt();
4. **Stopping a thread (I/O close)**

```java
public void run ()
{
    try
    {
        while (true)
        {
            byte [] buffer = inputStream.readBytes(4000);
            ... do something ...
        }
    } catch (IOException e)
    {
        .. clean up ...
    }
}
To stop the thread:
inputStream.close ();
```

5. **Coordinating threads with wait and notify**

**Producer thread**

```java
public synchronized void put(int value)
{
    ... make value available ...
    available = true;
    notify();
}
```

**Consumer thread**

```java
public synchronized int get()
{ while (available == false)
    { try
        { // wait for Producer to put value
            wait();
        } catch (InterruptedException e) { } }
    available = false;
    return contents;
}
```

6. **Manipulating a Swing interface from within the event queue**

```java
EventQueue.invokeLater(
    new Runnable()
    {
        public void run()
        {
            ... modify a UI component ...
        }
    });
```