Some notes on threads:

1. Creating a Thread

When a thread is created, it must be permanently bound to an object with a run() method. When the thread is started, it will invoke the object's run() method. More specifically, the object must implement the Runnable interface. There are two ways to create a thread.

The first is to declare a class that extends Thread. When the class is instantiated, the thread and object are created together and the object is automatically bound to the thread. By calling the object's start() method, the thread is started and immediately calls the object's run() method. Here is some code to demonstrate this method.

```java
// This class extends Thread
class BasicThread1 extends Thread {
    // This method is called when the thread runs
    public void run() {
    }
}

// Create and start the thread
Thread thread = new BasicThread1();
thread.start();
```

The second way is to create the thread and supply it an object with a run() method. This object will be permanently associated with the thread. The object's run() method will be invoked when the thread is started. This method of thread creation is useful if you want many threads sharing an object. Here is an example that creates a Runnable object and then creates a thread with the object.

```java
class BasicThread2 implements Runnable {
    // This method is called when the thread runs
    public void run() {
    }
}

// Create the object with the run() method
Runnable runnable = new BasicThread2();

// Create the thread supplying it with the runnable object
Thread thread = new Thread(runnable);

// Start the thread
thread.start();
```

2. Stopping a Thread

The proper way to stop a running thread is to set a variable that the thread checks occasionally. When the thread detects that the variable is set, it should return from the run() method.

Note: Thread.suspend() and Thread.stop() provide asynchronous methods of stopping a thread. However, these methods have been deprecated because they are very unsafe. Using them often results in deadlocks and incorrect resource cleanup.

```java
// Create and start the thread in some class
MyThread thread = new MyThread();
thread.start();

// Do work...in the original class

// Stop the thread
thread.allDone = true; //this is the Boolean in the class MyThread
```

```java
class MyThread extends Thread {
    boolean allDone = false;

    // This method is called when the thread runs
    public void run() {
        while (true) {
            // Do work...
            if (allDone) {
                return;
            }

            // Do work...
        }
    }
}
```
3. Pausing a Thread

The proper way to temporarily pause the execution of another thread is to set a variable that the target thread checks occasionally. When the target thread detects that the variable is set, it calls Object.wait(). The paused thread can then be woken up by calling its Object.notify() method.

Note: Thread.suspend() and Thread.resume() provide methods for pausing a thread. However, these methods have been deprecated because they are very unsafe. Using them often results in deadlocks. With the approach above, the target thread can ensure that it will be paused in an appropriate place.

```java
// Create and start the thread
MyThread thread = new MyThread();
thread.start();

while (true) {
    // Do work

    // Pause the thread
    synchronized (thread) {
        thread.pleaseWait = true;
    }

    // Do work

    // Resume the thread
    synchronized (thread) {
        // note the use of synchronized, this controls the access to pleaseWait variable
        thread.pleaseWait = false;
        thread.notify();
    }

    // Do work
}

class MyThread extends Thread {
    boolean pleaseWait = false;

    // This method is called when the thread runs
    public void run() {
        while (true) {
            // Do work

            // Check if should wait
            synchronized (this) {
                while (pleaseWait) {
                    try {
                        wait();
                    } catch (Exception e) {
                    }
                }

                // Do work
            }
        }
    }
}
```