```java
package thread_ex;
public class Thread_ex {

	boolean done = false;

	public Thread_ex(int NumT1, int NumT2) {

		/*
		 * Constructor for the main class, its function is to spawn the
		 * T1 and T2 threads
		 */
		for (int i = 0; i < NumT1; i++) {
			new T1().start();
		}
		for (int i = 0; i < NumT2; i++) {
			new T2().start();
		}

	}

	public class T1 extends Thread {
		/*
		 * T1 sleeps until done
		 */
		public void run() {
			System.out.println ("T1 thread started");
			while (!done) {
				try {

					Thread.sleep(1000);
				} catch (InterruptedException ex) {
					System.out.println("Interrupt in T1");
				}

			}
			System.out.println ("T1 thread finished");
		}
	}

	public class T2 extends Thread {
		/*
		 * T1 sleeps until done
		 */
		public void run() {
			System.out.println ("T2 thread started");
			while (!done) {
				try {

					Thread.sleep(1000);
				} catch (InterruptedException ex) {
					System.out.println("Interrupt in T2");
				}

			}
			System.out.println ("T2 thread finished");
		}

	}
	public static void main(String[] args) {
		Thread_ex tex = new Thread_ex(2,2);
		/*
		 * Sleep for 5 seconds and kill
		 */
		try {
			Thread.sleep(5000);
		} catch (InterruptedException e) {

		}
		tex.done = true;
		System.out.println("main finished");
	}

}
```

```
T1 thread started
T1 thread started
T2 thread started
T2 thread started
main finished
T2 thread finished
T2 thread finished
T1 thread finished
T1 thread finished
```

CS370 Worksheet 7 Name_____

1) What are the advantages of the one-to-one user/kernel thread model? Disadvantages? How about the many-to-many model?

2) What are the 4 options available for handling signals delivered to a multi-threaded process? When might you use them?

3) What are the two types of cancellation for threads? When might you use them?

4) What is the issue with fork() and exec()? How can it be handled?