The purpose of this assignment is to learn about multithreading in programming languages such as ADA and Java. More specifically, you should develop a multithreaded application in Java. Examine the following link on our course web site: http://www.iusb.edu/~hhakimza/311/java/ You will find a number of sample Java programs:

HelloWorldApp.java
Welcome4.java
Comparison.java
Copy.java
LabelTest.java
Painter.java
RadioButtonTest.java
ThreadTester.java

The first step is to compile and run each of the above programs using a Java compiler such as Sun’s J2E 1.3.1. Using the Sun Java compiler, these programs can be compiled via the `javac` command, and the resulting byte-code can be executed/interpreted using the `java` command (Java virtual machine).

```
javac HelloWorldApp.java  // this will generate one or more .class files
java HelloWorldApp.class   // this is the class file that has the main() function in it.
```

**Note:** When using the Java virtual machine, the class name “HelloWorldApp” should be typed as typed in the class specification (i.e. it is case sensitive!!)

After examining, compiling and running all the programs. Closely look at the ThreadTester.java program. This program will be the foundation of your assignment. Read and understand this multithreaded program. The main function of this program is to simply creates four Print_Thread objects and later calls the `start()` method of each thread. (See table 1)

Once the `start()` method is executed, the new thread has its own life and will execute independently. The `start()` function call is different from a traditional function calls. The `start()` simply places the new thread in to a ready/Runnable state (in the scheduling ring) and immediately returns to the main program. The newly created thread will now run on its own (See Table 2). The specifics behavior of threads in JAVA is environment dependent. (i.e. Preemptive vs. non-preemptive).
**Note:** Test the ThreadTester program in your environment and document its behavior before making any modifications to the code.

Once you completely understand the program, modify the ThreadTester program so that each thread does something more interesting than just sleeping for a while. For example, you can modify the program such that, each thread will be able to read and display a file. (Perhaps 1 line at a time) This would be a good example of the effect of multitasking if operating systems did not provide you with print spoolers. (You should get interleaved results!!)

Alternatively, you may use some of the other programs above! Note however, that there is no multiple inheritance in JAVA. Therefore plan accordingly. (Hint: Runnable Interface!!!)

```java
public class ThreadTester {
    // create and start threads
    public static void main( String args[] )
    {
        PrintThread thread1, thread2, thread3, thread4;

        // create four PrintThread objects
        thread1 = new PrintThread( "thread1" );
        thread2 = new PrintThread( "thread2" );
        thread3 = new PrintThread( "thread3" );
        thread4 = new PrintThread( "thread4" );

        System.err.println( "nStarting threads" );

        // start executing PrintThreads
        // Each new thread is started and the program returns to execute the following line.
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();

        System.err.println( "Threads started
" );
    }
}
```
Table 2

/**************************************************************************
* (C) Copyright 2002 by Deitel & Associates, Inc. and Prentice Hall.     *
* All Rights Reserved.                                                   *
*                                                                        *
**************************************************************************/

// Each object of this class picks a random sleep interval.
// When a PrintThread executes, it prints its name, sleeps,
// prints its name again and terminates.

class PrintThread extends Thread {
    private int sleepTime;

    // PrintThread constructor assigns name to thread
    // by calling superclass Thread constructor
    public PrintThread( String name )
    {
        super( name );

        // sleep between 0 and 5 seconds
        sleepTime = (int) ( Math.random() * 5000 );

        // display name and sleepTime
        System.err.println(
            "Name: " + getName() + ";  sleep: " + sleepTime );
    }

    // control thread's execution
    public void run()
    {
        // put thread to sleep for a random interval
        try {
            System.err.println( getName() + " going to sleep" );

            // put thread to sleep
            Thread.sleep( sleepTime );
        }

        // if thread interrupted during sleep, catch exception
        // and display error message
        catch ( InterruptedException interruptionException ) {
            System.err.println( interruptionException.toString() );
        }

        // print thread name
        System.err.println( getName() + " done sleeping" );
    }

} // end class PrintThread

Keep in mind that your program should document what it hopes to achieve and
effectively demonstrate the result of multitasking.
Note: For this assignment, you have the option to work with up to two other classmates and form a team. If you choose to do so, each member should be able to fully articulate the solution adopted/developed by the team. Also, each member should document what aspect of the design and implementation they worked on individually.

What to hand in?

1) A cover page with the usual information
2) Discussion of your solution
3) Program Listing (fully documented)
4) Sample output (annotated if necessary)
5) Staple your paper. (No loose papers please)