JAVA HW 10.1

```java

public class ThreadQuestionOne extends Thread    {
    private  static Random random = new Random();
    String info;

    public ThreadQuestionOne (String info) {
        this.info     = info;
    }

     public void run () {
         System.err.println(info + " ---> ");
         System.err.println(info + " <--- ");

         if ( info == "first" )
                 new ThreadQuestionOne("fourth").start();
     }

     public static void main (String args []) {
         ThreadQuestionOne aT4_0 = new ThreadQuestionOne("first");
         ThreadQuestionOne aT4_1 = new ThreadQuestionOne("second");
         ThreadQuestionOne aT4_2 = new ThreadQuestionOne("third");

         aT4_0.run();
         aT4_1.run();
         aT4_2.start();
     }
 }
```

1. After execution, firstly an object is created named “aT4_0” which keeps its info string to “first”.
2. Then a second object is created named “aT4_1” which keeps its info string as “second”.
3. And lastly, a third object is created named “aT4_2” which keeps its info string as “third”.
4. After that, the run method is called on the aT4_0 object which prints first ---> and then first <---. These two statements are always together here because there is no thread created till now.
5. After that, the info is equal to “first” so a new thread is ready to run on a new object. Now there are two threads ready to run, the main thread and the fourth thread.
6. Now if the main thread runs, then the method run is called on the second object. And it can print the output, second —> second <---
7. After this, a new thread is created for the third object, now since we have three threads ready to run, namely, main, fourth and third, so the output can be jumbled up anyhow but the right arrow always comes before the left arrow, be it any object.
8. So, there can be any pattern next with the second, third and fourth object, the only need being the right arrow will always be first and then the left arrow comes.

```
first --->
fourth --->
first <---
second --->
fourth <---
third --->
second <---
third <---
```

This output is not possible because the first ---> and the first <--- is separated by the fourth --->, which is not possible.