I have had a fresh attempt. I have used my failed attempt to step up the learning curve further.. I paid much more emphasis straight away on identifying if the merge is actually valid before trying to progress anything further....

I have implemented lots of code and the main problem so far is this similar code which is appearing in multiple sections of my code.
NOTE: It just not has been possible to walkthrough my implementations since it was extremely intertwined..

For now, I have acknowledged it is repeat code and not concerned about redundancy..
I can see clearly see there is a common trend between variables which require adjustment... It is bespoke to each section of the code in which it appears, so it pointed me towards creating a method...
Since the datatypes are the same for arguments and parameters, I can not use any method overloading...

But it requires extra close attention to ensure that correct information is passed in, since it ascertains if the merged interval is based on ascending [3,5],[4,6]

Unfortunately due to the structure of my code, I have ended up in this section since I created my code in two halves... (in which counter>2 and counter<=2).. This was in reference to the objects extracted from the list.
So effectively, I had to provision similar method in both sides of the code.

This is the repeat code that appears in three areas...
I have highlighted the variables which differ on each section... This will assist in passing correct values...

Please note, also in each section, I am outputting the following variables on the screen. Note that all are not necessarily initialised correctly depending on the situation..
I am using this along with my other screen outputs to make sense of situation.

```
System.out.println("start interval: " + startInterval);
System.out.println("end interval: " + endInterval);
System.out.println("start interval second range: " + startIntervalSecondRange);

System.out.println("end interval second range: " + endIntervalSecondRange);

System.out.println("start interval first range: " + startIntervalFirstRange);

System.out.println("end interval first range: " + endIntervalFirstRange);
```

REPEAT CODE:

I have colour coded so that it can be seen which variables pass into the area...
The downfall on code consolidation is that it will be time consuming trying to study the code again in future... So once all is functional, I will endeavour to keep both versions....

```
 //This classifies as a removal from ascending perspective     or descending perspective

   if ((Integer.valueOf(endIntervalFirstRange)>=Integer.valueOf(endInterval
startIntervalSecondRange)  &&
Integer.valueOf(endIntervalSecondRange)>=Integer.valueOf(startInterval
startIntervalSecondRange))

   ||  (Integer.valueOf(endIntervalFirstRange)<=Integer.valueOf(endInterval
startIntervalSecondRange)  &&
Integer.valueOf(endIntervalSecondRange)<=Integer.valueOf(startInterval
startIntervalSecondRange)))

                {

                   System.out.println("1Removed last item from innerList: " +
innerList.get(innerList.size()-1));

                innerList.remove(innerList.size()-1);


                newInterval = "["+startInterval  startIntervalFirstRange +","+ " " +
endIntervalFirstRange  endIntervalSecondRange +"]MERGED";

                innerList.add(newInterval);

                System.out.println("3Added into Inner List: " + newInterval);

                System.out.println("This is innerlist: " + innerList);


                }


                //}

                //it would simply have no merging even though the ranges overlap

                else

                {
```

```
            newInterval = "["+startIntervalFirstRange  startIntervalSecondRange +","+ " "
+ endIntervalFirstRange  endIntervalSecondRange +"]NO MERGE";

            innerList.add(newInterval);

            System.out.println("2Added into Inner List: " + newInterval);

            }

            ---------------------------------------------------------------------------
```

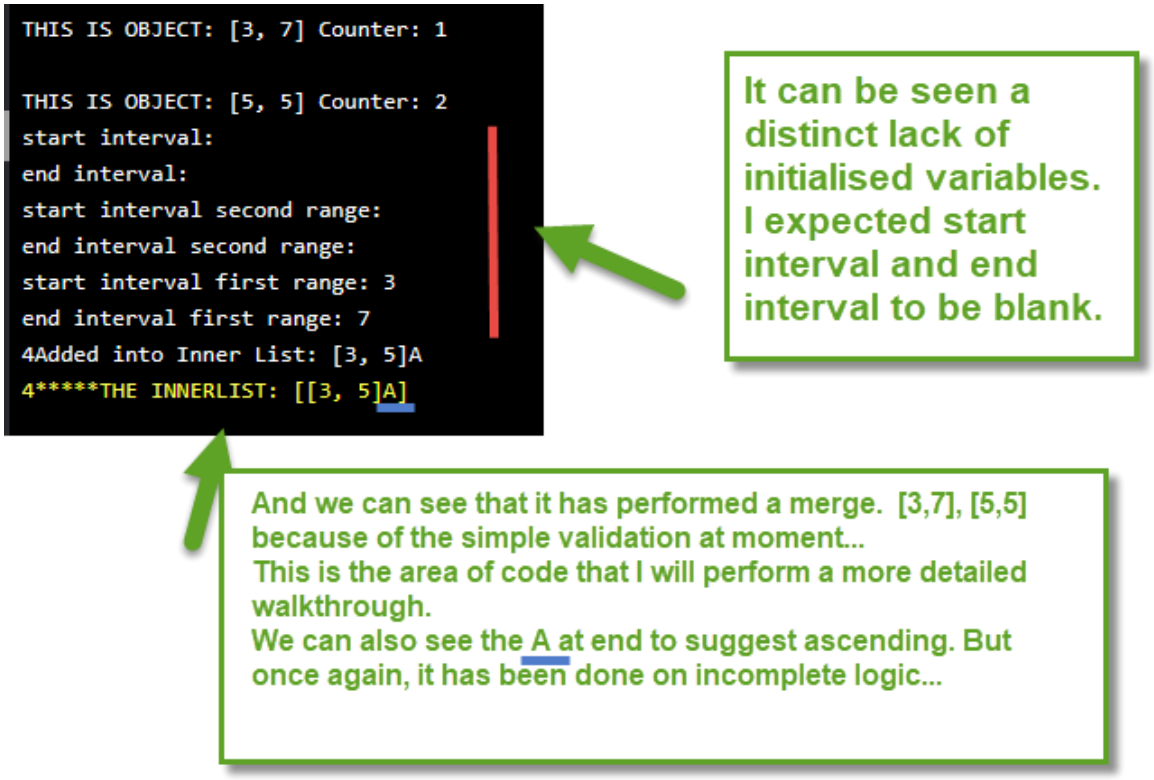
For now, I will study the third instance in which the method above is required...
I will just copy the whole original code and fill the variable used in a different colour..
I can then make a late decision as to whether I need to consolidate.

TEST CASE:

```
//TEST CASE 20 => using ascending and descending but matching intervals (****FAIL******)
entries.add("[3, 7]");
entries.add("[5, 5]");
entries.add("[4, 3]");
```

At the moment, the logic is not instilled in this area of code to make decision on [5,5]. It is practically in same state as the version that passed with Programiz. Note in Programiz, there were no validation checks required since the test cases were expected to fit in the scenario prescribed (merging ascending).

I have examined my screen output and it has hence performed,



I have run into endless issues, the more I tried to remember my code, the more I was forgetting it.
But I definitely knew the framework was correct...
See the testing undertaken at top of code, I will just try lots scenarios on it...

I have just developed a single version with all the code in the single non-main method.
Anything else would make this a disaster to remediate..