I have now reached the position in which I can run test cases.
Unfortunately with this challenge, I just could not create different versions of code (upon having figured out how to navigate the inner and outer Lists).

The changes towards the end required changes in several places, I had to exhaust the level of debugging.

And I could see eventually I was getting closer to test scenarios.. But unfortunately all were not passing.

TEST CASE 1: **("{1,3}, {2,8}, {6,10}, {11,14}");** **PASS**

//PASS - all overlap except last entry

System.out.println("{1,3}, {2,8}, {6,10}, {9,14}");

entries.add("[1, 3]");

entries.add("[2, 8]");

entries.add("[6, 10]");

entries.add("[9, 14]");

****OUTPUT**********

{1,3}, {2,8}, {6,10}, {11,14}

REACH HERE

This is the object: [1, 3]

This is counter: 1

RANGE1 start:1

RANGE1 end:3

[1, 3]

3

CHANCE BACK HERE

This is the object: [2, 8]

This is counter: 2

RANGE2 start:2

RANGE2 end:8

[2, 8]

[1, 3]

3

2

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 4

**********New interval into outer List ADDED :[1, 8]

CHANCE BACK HERE

counter being reset

This is the object: [6, 10]

This is counter: 1

RANGE1 start:6

RANGE1 end:10

[2, 8]

[6, 10]

10

2

CHANCE BACK HERE

this is current new interval: [1, 8]

[[1, 8]]

REVISED ENDNEWINTERVAL: 8

REVISED startinterval: 6

**********OVERLAP:

[2, 8]

[6, 10]

ultimate start of range: 1

**********REMOVED: [1, 8]

**********NEW INTERVAL SET ADDED => [1, 10]

This is the object: [11, 14]

This is counter: 2

RANGE2 start:11

RANGE2 end:14

[11, 14]

[6, 10]

10

11

UUUUUUUU

HERE1

----------DATA CHECK

10

10

111INNER list right now: [[1, 10]]

IS THERE OVERLAP: false

10

10

789**********ADDED: [11, 14]

INNER list right now: [[1, 10]]

CHANCE BACK HERE

counter being reset

checking

THIS IS THE OUTER LIST: [[[1, 10], [11, 14]]]

//ALTHOUGH NOT FORMATTED CORRECTLY, I WAS CONTENT TO LET THIS PASS SINCE IT WAS POPULATED AS EXPECTED

** Process exited - Return Code: 0 **

TEST CASE 2: ("{1,3}, {2,8}, {6,10}, {9,14} **PASS**

```
//PASS    - all overlap
        System.out.println("{1,3}, {2,8}, {6,10}, {9,14}");
        entries.add("[1, 3]");
        entries.add("[2, 8]");
        entries.add("[6, 10]");
        entries.add("[9, 14]");
```

{1,3}, {2,8}, {6,10}, {9,14}

REACH HERE

This is the object: [1, 3]

This is counter: 1

RANGE1 start:1

RANGE1 end:3

[1, 3]

3

CHANCE BACK HERE

This is the object: [2, 8]

This is counter: 2

RANGE2 start:2

RANGE2 end:8

[2, 8]

[1, 3]

3

2

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 4

**********New interval into outer List ADDED :[1, 8]

CHANCE BACK HERE

counter being reset

This is the object: [6, 10]

This is counter: 1

RANGE1 start:6

RANGE1 end:10

[2, 8]

[6, 10]

10

2

CHANCE BACK HERE

this is current new interval: [1, 8]

[[1, 8]]

REVISED ENDNEWINTERVAL: 8

REVISED startinterval: 6

**********OVERLAP:

[2, 8]

[6, 10]

ultimate start of range: 1

**********REMOVED: [1, 8]

**********NEW INTERVAL SET ADDED => [1, 10]

This is the object: [9, 14]

This is counter: 2

RANGE2 start:9

RANGE2 end:14

[9, 14]

[6, 10]

10

9

UUUUUUUU

OVERLAP

Object number: 4

Objects in intervals: 4

BREAKING OUT

this is counter: 2

CHANCE BACK HERE

this is current new interval: [1, 10]

[[1, 10]]

REVISED ENDNEWINTERVAL: 10

REVISED startinterval: 9

**********OVERLAP:

[9, 14]

[6, 10]

ultimate start of range: 1

**********REMOVED: [1, 10]

**********NEW INTERVAL SET ADDED => [1, 14]

checking

THIS IS THE OUTER LIST: [[[1, 14]]]

** Process exited - Return Code: 0 **

At this point I was extremely satisfied

//PASS - all overlap except last entry TEST CASE 1: **("{1,3}, {2,8}, {6,10}, {11,14}");**
// ALL OVERLAP NOW - PASS TEST CASE 2: ("{1,3}, {2,8}, {6,10}, {11,14}

But I knew there would be difficulties ahead. For now I just simplified the test case

TEST CASE 3: ("{1,3}, {5,8} **PASS**

// NO OVERLAP NOW (less objects) - PASS

```
                System.out.println("{1,3}, {5,8}");
                  entries.add("[1, 3]");
                  entries.add("[5, 8]");
```

{1,3}, {5,8}

REACH HERE

This is the object: [1, 3]

This is counter: 1

RANGE1 start:1

RANGE1 end:3

[1, 3]

3

CHANCE BACK HERE

This is the object: [5, 8]

This is counter: 2

RANGE2 start:5

RANGE2 end:8

[5, 8]

[1, 3]

3

5

UUUUUUUU

HERE1

----------DATA CHECK

3

111INNER list right now: []

IS THERE OVERLAP: false

3

123**********ADDED: [1, 3]

125**********ADDED: [5, 8]

INNER list right now: [[1, 3], [5, 8]]

CHANCE BACK HERE

counter being reset

checking

THIS IS THE OUTER LIST: [[[1, 3], [5, 8]]]

** Process exited - Return Code: 0 **

TEST CASE 4: ("{1,3}, {2,8} FAIL

```
  // OVERLAP NOW (less objects) - PASS
                    System.out.println("{1,3}, {2,8}");
                      entries.add("[1, 3]");
                      entries.add("[2, 8]");
```

{1,3}, {2,8}

REACH HERE

This is the object: [1, 3]

This is counter: 1

RANGE1 start:1

RANGE1 end:3

[1, 3]

3

CHANCE BACK HERE

This is the object: [2, 8]

This is counter: 2

RANGE2 start:2

RANGE2 end:8

[2, 8]

[1, 3]

3

2

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 2

BREAKING OUT

this is counter: 2

CHANCE BACK HERE

this is current new interval: //we can see this is empty. As a reminder, we can see the new interval is calculated as follows (see photo below)

[]

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin 0, end -1, length 0

at java.base/java.lang.String.checkBoundsBeginEnd(String.java:3319)

at java.base/java.lang.String.substring(String.java:1874)

at Solution.mergeIntervals(Solution.java:298)

at Solution.main(Solution.java:68)

** Process exited - Return Code: 1 **

```
if (numberObjects==objectNumber)
{
    //innerList.add(even);
    System.out.println("BREAKING OUT");
    System.out.println("this is counter: " + counter);
    counter=1;

}

if (counter==2)
{
 newInterval = "["+oddSubStringStartInterval+", "+evenSubStringEndInterval+"]";
 innerList.add(newInterval);
 System.out.println("**********************************************New interval int
 overlap=true;
```

However it can be seen that since object in intervals is equal to object number, my code sets counter=1 and therefore it can not configure a newInterval.....

```
293          //this seems to be more straightforward...
294          // if it does not overlap, it will set the flag to false....
295
296         System.out.println("this is current new interval:  " + newInterval);
297         System.out.println(innerList);
298
299          endNewInterval = newInterval.substring(newInterval.indexOf(" ")+1,newInterval.indexOf(
300          System.out.println("REVISED ENDNEWINTERVAL: " + endNewInterval);
301
302          String subStringStartInterval = String.valueOf(temp).substring(1,even.indexOf(","));
303          System.out.println("REVISED startinterval: " + subStringStartInterval);
304
```

To fix this, I can simply put logic as follows....
if (newInterval=="")
Add the odd and even String into the innerList
Then break;
However it would store both entries separately.... even though overlapping ranges

So this leaves me with an issue, rather than adding more code for this scenario (which seems illogical) since I already had the scenario running when counter==2
I have to get rid of the condition in the first photo and see impact across all my test cases again!

TEST CASE 4a: removed counter=1 from first photo **PASS**

PASS

{1,3}, {2,8}

REACH HERE

This is the object: [1, 3]

This is counter: 1

RANGE1 start:1

RANGE1 end:3

[1, 3]

3

CHANCE BACK HERE

This is the object: [2, 8]

This is counter: 2

RANGE2 start:2

RANGE2 end:8

[2, 8]

[1, 3]

3

2

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 2

BREAKING OUT

this is counter: 2

**********New interval into outer List ADDED :[1, 8]

CHANCE BACK HERE

counter being reset

checking

THIS IS THE OUTER LIST: [[[1, 8]]]

** Process exited - Return Code: 0 **

However I am now expecting all test cases before TEST CASE 4 to fail.
I will go through them individually

TEST CASE 1: **("{1,3}, {2,8}, {6,10}, {11,14}"); - revisited = PASS previously = PASS**

TEST CASE 2: ("{1,3}, {2,8}, {6,10}, {9,14} revisited = FAIL previously = **PASS**

CHANCE BACK HERE

counter being reset

checking

THIS IS THE OUTER LIST: [[[1, 10], [6, 14]]]

TEST CASE 3: ("{1,3}, {5,8} - revisited = PASS previously = PASS

THIS IS THE OUTER LIST: [[[1, 3], [5, 8]]]

TEST CASE 4: Already resolved with 4a

TEST CASE 5: ("{1,5}, {4,9}, {12,17}, {20,25}"); = Still fails FAIL

TEST CASE 6 ("{1,5}, {6,9}, {12,17}, {20,25}"); = still crashes FAIL

TEST CASE 7 ("{1,5}, {6,9}, {8,17}, {16,25}"); = still crashes FAIL

I can see straight away I have not been any better off so I have re-instated the counter=1

I have also realised I have not tried most basic testing of two ranges...
with all permutations possible

TEST CASE WITH TWO RANGES PERMUTATIONS:

System.out.println("{1,3}, {2,8}"); = CRASH (TEST CASE 4)... = NOW FIXED....
System.out.println("{1,4}, {5,7}") = PASS

I am now shifting my priorities on these scenarios since it is even failing with three ranges..

System.out.println("{1,4}, {2,6}, {7,9}") = FAIL
System.out.println("{1,4}, {5,7}, {8,9}") = CRASH

TEST CASE 4b ("{1,3}, {2,8} = NOW PASSES **PASS**

Unfortunately I might need an approach now so that it performs the intrinsic boundary check. I added following code and now it passes.
I do not expect this to break any other tests since it is only test with two objects.

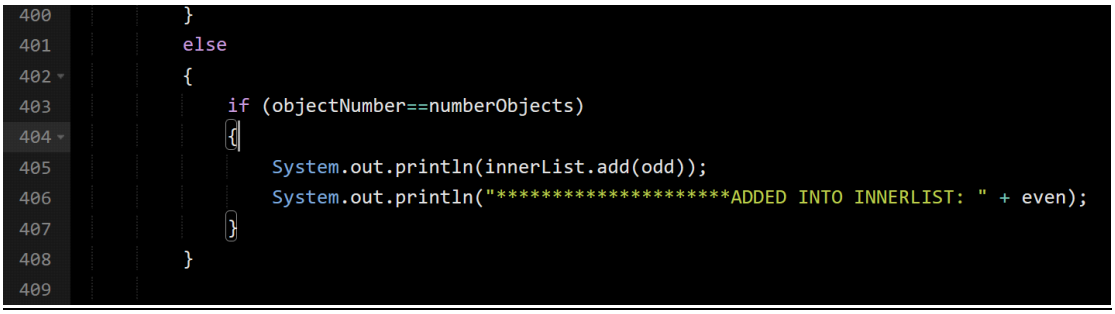
```
186
187             if (numberObjects==2)
188             {
189                 newInterval = "["+oddSubStringStartInterval+", "+evenSubStringEndInterval+"]";
190                  innerList.add(newInterval);
191                  break;
192
193             }
194
```

My focus is now on this:

TEST CASE 4c {1,4}, {2,6}, {7,9} FAIL = Giving [1,6]

I have to understand why it fails to add [7,9] **PASS with changes**

I added the following code (if statement):



```
400         }
401         else
402         {
403             if (objectNumber==numberObjects)
404             {
405                 System.out.println(innerList.add(odd));
406                 System.out.println("********************ADDED INTO INNERLIST: " + even);
407             }
408         }
409
```

And now it functions:

TEST CASE 4d {1,4}, {5,7}, {8,9} FAIL = Giving [1,6] - This now crashes...

INNER list right now: [[1, 4], [5, 7]] = **PASS**

CHANCE BACK HERE

counter being reset

This is the object: [8, 9](Object number: 3)

This is counter: 1

RANGE1 start:8

RANGE1 end:9

[5, 7] //this will be odd String

[8, 9] //this will be even String

9

5

CHANCE BACK HERE

this is current new interval: //there is no value

[[1, 4], [5, 7]]

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin 0, end -1, length 0

at java.base/java.lang.String.checkBoundsBeginEnd(String.java:3319)

at java.base/java.lang.String.substring(String.java:1874)

at Solution.mergeIntervals(Solution.java:323)

at Solution.main(Solution.java:72)

I can see the following in red.
I can follow same principle as 4c and add code into anywhere before line 323
It now passes

I tried a few more test cases which all fail.....

TEST CASE 5: ("{1,5}, {4,9}, {12,17}, {20,25}"); FAIL

```
//FAILS - one overlap      it gives [1,17],    [20,25]

                System.out.println("{1,5}, {4,9}, {12,17}, {20,25}");

                  entries.add("[1, 5]");

                  entries.add("[4, 9]");

                  entries.add("[12, 17]");

                  entries.add("[20, 25]");
```

{1,5}, {4,9}, {12,17}, {20,25}

REACH HERE

This is the object: [1, 5]

This is counter: 1

RANGE1 start:1

RANGE1 end:5

[1, 5]

5

CHANCE BACK HERE

This is the object: [4, 9]

This is counter: 2

RANGE2 start:4

RANGE2 end:9

[4, 9]

[1, 5]

5

4

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 4

**********New interval into outer List ADDED :[1, 9]

CHANCE BACK HERE

counter being reset

This is the object: [12, 17]

This is counter: 1

RANGE1 start:12

RANGE1 end:17

[4, 9]

[12, 17]

17

4

CHANCE BACK HERE

this is current new interval: [1, 9]

[[1, 9]]

REVISED ENDNEWINTERVAL: 9

REVISED startinterval: 1

**********OVERLAP:

[4, 9]

[12, 17]

ultimate start of range: 1

**********REMOVED: [1, 9]

**********NEW INTERVAL SET ADDED => [1, 17]

This is the object: [20, 25]

This is counter: 2

RANGE2 start:20

RANGE2 end:25

[20, 25]

[12, 17]

17

20

UUUUUUUU

HERE1

----------DATA CHECK

17

17

111INNER list right now: [[1, 17]]

IS THERE OVERLAP: false

17

17

789**********ADDED: [20, 25]

INNER list right now: [[1, 17]]

CHANCE BACK HERE

counter being reset

checking

THIS IS THE OUTER LIST: [[[1, 17], [20, 25]]]

** Process exited - Return Code: 0 **

TEST CASE 6 ("{1,5}, {6,9}, {12,17}, {20,25}"); FAIL

//FAILS - no overlap. it adds first two then crashes...

```
System.out.println("{1,5}, {6,9}, {12,17}, {20,25}");
  entries.add("[1, 5]");
  entries.add("[6, 9]");
  entries.add("[12, 17]");
  entries.add("[20, 25]");
```

{1,5}, {6,9}, {12,17}, {20,25}

REACH HERE

This is the object: [1, 5]

This is counter: 1

RANGE1 start:1

RANGE1 end:5

[1, 5]

5

CHANCE BACK HERE

This is the object: [6, 9]

This is counter: 2

RANGE2 start:6

RANGE2 end:9

[6, 9]

[1, 5]

5

6

UUUUUUUU

HERE1

----------DATA CHECK

5

111INNER list right now: []

IS THERE OVERLAP: false

5

123**********ADDED: [1, 5]

125**********ADDED: [6, 9]

INNER list right now: [[1, 5], [6, 9]]

CHANCE BACK HERE

counter being reset

This is the object: [12, 17]

This is counter: 1

RANGE1 start:12

RANGE1 end:17

[6, 9]

[12, 17]

17

6

CHANCE BACK HERE

this is current new interval:

[[1, 5], [6, 9]]

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin 0, end -1, length 0

at java.base/java.lang.String.checkBoundsBeginEnd(String.java:3319)

at java.base/java.lang.String.substring(String.java:1874)

at Solution.mergeIntervals(Solution.java:299)

at Solution.main(Solution.java:69)

** Process exited - Return Code: 1 **

TEST CASE 7 ("{1,5}, {6,9}, {8,17}, {16,25}"); FAIL

```
//FAILS - no overlap first two then overlap then overlap
                //it adds first two into innerList then crashes
                System.out.println("{1,5}, {6,9}, {8,17}, {16,25}");
                  entries.add("[1, 5]");
                  entries.add("[6, 9]");
                  entries.add("[8, 17]");
                  entries.add("[16, 25]");
```

{1,5}, {6,9}, {8,17}, {16,25}

REACH HERE

This is the object: [1, 5]

This is counter: 1

RANGE1 start:1

RANGE1 end:5

[1, 5]

5

CHANCE BACK HERE

This is the object: [6, 9]

This is counter: 2

RANGE2 start:6

RANGE2 end:9

[6, 9]

[1, 5]

5

6

UUUUUUUU

HERE1

----------DATA CHECK

5

111INNER list right now: []

IS THERE OVERLAP: false

5

123**********ADDED: [1, 5]

125**********ADDED: [6, 9]

INNER list right now: [[1, 5], [6, 9]]

CHANCE BACK HERE

counter being reset

This is the object: [8, 17]

This is counter: 1

RANGE1 start:8

RANGE1 end:17

[6, 9]

[8, 17]

17

6

CHANCE BACK HERE

this is current new interval:

[[1, 5], [6, 9]]

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: begin 0, end -1, length 0

    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:3319)

    at java.base/java.lang.String.substring(String.java:1874)

    at Solution.mergeIntervals(Solution.java:299)

    at Solution.main(Solution.java:69)

** Process exited - Return Code: 1 **

I have completed a decent level of debugging code... So I am hoping I can start with the most basic failing cases... And then try to figure out the more complex ones!!!

So now it appears, I need to resolve following test cases (2,5,6,7)

TEST CASE 2: ("{1,3}, {2,8}, {6,10}, {9,14} = NOW PASSES **PASS**

This code broken with the changes, but now it is better to move forward and seek remediations.
THIS IS THE OUTER LIST: [[[1, 10], [6, 10]]]
I need to be careful now since I do not want to disrupt the other test cases.
I have changed this from odd => even
I am certain it will break another test case... I will quickly go through all of them again (Test case 1 = PASS, TEST CASE 3 = PASS TEST CASE 4 = PASS
TEST CASE 4C NOW FAILS ({1,4}, {2,6}, {7,9}),
TEST CASE 4D NOW FAILS..... {1,4}, {5,7}, {8,9}

```
322
323         if (objectNumber==numberObjects)
324             {
325                 System.out.println(innerList.add(even));
326                 System.out.println("***CHANGE*******************ADDED INTO INNERLIST: " + even
327                 break;
328             }
329
```

So I have reversed the above code back and here are outputs so that I can analyse situation.

TEST CASE 4C OUTPUT **PASS**

{1,4}, {2,6}, {7,9}

REACH HERE

This is the object: [1, 4](Object number: 1)

This is counter: 1

RANGE1 start:1

RANGE1 end:4

[1, 4]

4

CHANCE BACK HERE

This is the object: [2, 6](Object number: 2)

This is counter: 2

RANGE2 start:2

RANGE2 end:6

[2, 6]

[1, 4]

4

2

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 3

**********New interval into outer List

ADDED :[1, 6]

CHANCE BACK HERE

counter being reset

This is the object: [7, 9](Object number: 3)

This is counter: 1

RANGE1 start:7

RANGE1 end:9

[2, 6] //this is even

[7, 9] //this is odd String

9

2

CHANCE BACK HERE

this is current new interval: [1, 6] //it has identified the current interval

[[1, 6]]

true

CHANGE*******ADDED INTO INNERLIST: [7, 9]

checking

THIS IS THE OUTER LIST: [[[1, 6], [7, 9]]]

** Process exited - Return Code: 0 **
This seems all fine

TEST CASE 4D OUTPUT **PASS**

{1,4}, {5,7}, {8,9}

REACH HERE

This is the object: [1, 4](Object number: 1)

This is counter: 1

RANGE1 start:1

RANGE1 end:4

[1, 4]

4

CHANCE BACK HERE

This is the object: [5, 7](Object number: 2)

This is counter: 2

RANGE2 start:5

RANGE2 end:7

[5, 7]

[1, 4]

4

5

UUUUUUUU

HERE1

----------DATA CHECK

4

111INNER list right now: []

IS THERE OVERLAP: false

4

123**********ADDED: [1, 4]

125**********ADDED: [5, 7]

INNER list right now: [[1, 4], [5, 7]]

CHANCE BACK HERE

counter being reset

This is the object: [8, 9](Object number: 3)

This is counter: 1

RANGE1 start:8

RANGE1 end:9

[5, 7]

[8, 9] //this is odd String

9

5

CHANCE BACK HERE

this is current new interval:

[[1, 4], [5, 7]]

CHANGE*******ADDED INTO INNERLIST: [8, 9]

checking

THIS IS THE OUTER LIST: [[[1, 4], [5, 7], [8, 9]]]

** Process exited - Return Code: 0 **
AGAIN IT SEEMS OK

TEST CASE 2 OUTPUT (FAIL)

{1,3}, {2,8}, {6,10}, {9,14}

REACH HERE

This is the object: [1, 3](Object number: 1)

This is counter: 1

RANGE1 start:1

RANGE1 end:3

[1, 3]

3

CHANCE BACK HERE

This is the object: [2, 8](Object number: 2)

This is counter: 2

RANGE2 start:2

RANGE2 end:8

[2, 8]

[1, 3]

3

2

UUUUUUUU

OVERLAP

Object number: 2

Objects in intervals: 4

**********New interval into outer List

ADDED :[1, 8]

CHANCE BACK HERE

counter being reset

This is the object: [6, 10](Object number: 3)

This is counter: 1

RANGE1 start:6

RANGE1 end:10

[2, 8]

[6, 10]

10

2

CHANCE BACK HERE

this is current new interval: [1, 8]

[[1, 8]]

REVISED ENDNEWINTERVAL: 8

REVISED startinterval: 6

**********OVERLAP:

[2, 8]

[6, 10]

ultimate start of range: 1

**********REMOVED: [1, 8]

**********NEW INTERVAL SET ADDED => [1, 10]

This is the object: [9, 14](Object number: 4)

This is counter: 2

RANGE2 start:9

RANGE2 end:14

[9, 14] //this is even String

[6, 10] //this is odd String

10

9

UUUUUUUU

OVERLAP

Object number: 4

Objects in intervals: 4

BREAKING OUT

this is counter: 2

[1, 10]

CHANCE BACK HERE

this is current new interval: [1, 10]

[[1, 10]]

CHANGE*******ADDED INTO INNERLIST: [6, 10]
//we are looking for the value [9,14] to be added here... (even String)

checking

THIS IS THE OUTER LIST: [[[1, 10], [6, 10]]]

** Process exited - Return Code: 0 **

This suggests adding following logic.... if counter ==2 then store even String.
if counter==1 then store odd String...

I will explore this around this area of code....

```
322
323         if (objectNumber==numberObjects)
324             {
325                 System.out.println(innerList.add(even));
326                 System.out.println("***CHANGE**********************ADDED INTO INNERLIST: " + even
327                 break;
328             }
329
```

And then I will re-test Test Case 2, Test Case 4d and Test Case 4c

Test Case 2v1: {1,3}, {2,8}, {6,10}, {9,14} **PASS**

```
324     if (objectNumber==numberObjects)
325         {
326             if (counter ==1)
327             {
328             innerList.add(odd);
329             System.out.println("***CHANGE**********************ADDED INTO INNERLIST: " + odd);
330             break;
331             }
332             else
333             {
334             innerList.add(even);
335             System.out.println("***CHANGE**********************ADDED INTO INNERLIST: " + even);
336             break;
337
338             }
```

THIS IS THE OUTER LIST: [[[1, 6], [7, 9]]]

Test Case 4d
I expect no issues since it still has odd condition in above

Test Case 4c
I expect no issues since it still has odd condition in above

I will now move across to following test cases:

TEST CASE 5: ("{1,5}, {4,9}, {12,17}, {20,25}"); = FAIL

**********New interval into outer List ADDED :[1, 9]

CHANCE BACK HERE

counter being reset

This is the object: [12, 17](Object number: 3)

This is counter: 1

RANGE1 start:12

RANGE1 end:17

[4, 9]

[12, 17]

17

4

CHANCE BACK HERE

this is current new interval: [1, 9] **//this is perfectly fine**

[[1, 9]]

REVISED ENDNEWINTERVAL: 9

REVISED startinterval: 1

**********OVERLAP: **//this is complete wrong, no overlap. It should be comparing 12 against 9... I will investigate.**
//I can see this is the first time in my test cases where I have first two ranges that merge and the third range that will not merge.... So it is ending up in a different loop.....

[4, 9]

[12, 17]

ultimate start of range: 1

**********REMOVED: [1, 9]** **//this is complete wrong, no overlap**

**********NEW INTERVAL SET ADDED => [1, 17]**

TEST CASE 6 ("{1,5}, {6,9}, {12,17}, {20,25}"); = NOW PASSES **PASS**
//this is complete non-merging
Crash is due to now value in newInterval...
//So I adapted this code:

```
315
316                 if (newInterval=="")
317                 {
318                     newInterval = String.valueOf(innerList.get(innerList.size()-1));
319
320                     //break;
321                 }
322
```

Now outcome is:

THIS IS THE OUTER LIST: [[[1, 5], [6, 17], [20, 25]]]

We can see that it has merged 6-17 however this is incorrect.
I will try to investigate this aspect.

I spotted a quite fatal mistake in my code..
And it makes me wonder if it has affected any other test cases!
This should have been referencing temp.indexOf

```
String subStringStartInterval = String.valueOf(temp).substring(1,even.indexOf(","));
```

I have performed the correction and now the test case passes!!

```
348  String subStringStartInterval = String.valueOf(temp).substring(1,String.valueOf(temp).indexOf(","));
```

THIS IS THE OUTER LIST: [[[1, 5], [6, 9], [12, 17], [20, 25]]]

TEST CASE 7 ("{1,5}, {6,9}, {8,17}, {16,25}"); FAIL

previously = crashes now = executes

{1,5}, {6,9}, {8,17}, {16,25}

REACH HERE

This is the object: [1, 5](Object number: 1)

This is counter: 1

RANGE1 start:1

RANGE1 end:5

[1, 5]

5

CHANCE BACK HERE

This is the object: [6, 9](Object number: 2)

This is counter: 2

RANGE2 start:6

RANGE2 end:9

[6, 9]

[1, 5]

5

6

UUUUUUUU

HERE1

----------DATA CHECK

5

111INNER list right now: []

IS THERE OVERLAP: false

5

123**********ADDED: [1, 5]

125**********ADDED: [6, 9]

INNER list right now: [[1, 5], [6, 9]]

CHANCE BACK HERE

counter being reset

This is the object: [8, 17](Object number: 3)

This is counter: 1

RANGE1 start:8

RANGE1 end:17

[6, 9]

[8, 17]

17

6

CHANCE BACK HERE

this is current new interval:

[[1, 5], [6, 9]]

This has been assigned to new interval: [6, 9]

REVISED ENDNEWINTERVAL: 9

----CURRENTLY IN TEMP: [8, 17]

REVISED startinterval: 8

**********OVERLAP:

[6, 9]

[8, 17]

ultimate start of range: 6

**********REMOVED: [6, 9]

**********NEW INTERVAL SET ADDED => [6, 17]

This is the object: [16, 25](Object number: 4)

This is counter: 2

RANGE2 start:16

RANGE2 end:25

[16, 25] //even String

[8, 17] //odd String

17

16

UUUUUUUU

OVERLAP

Object number: 4

Objects in intervals: 4

BREAKING OUT

this is counter: 2 // I know in this section of code the counter will be set to 1. Hence it is picking up the odd String. I need to look at debugging logs for 4c and 4d and also 2 to see if this sort of activity occurred... And understand it from grand scheme of things....

My investigation: it changed counter==1 on TEST CASE 2
This has also affected its final outcome....

[6, 17]

CHANCE BACK HERE

this is current new interval: [6, 17]

[[1, 5], [6, 17]]

*****CHANGE********************ADDED INTO INNERLIST: [8, 17]**
//the issue is clearly here with the new multi-purpose implemented code.
//But I need to understand why it works for other situations such as Test case 4c and 4d and not this one...

checking

THIS IS THE OUTER LIST: [[[1, 5], [6, 17], **[8, 17]]]**
// I can see that [8,17] is incorrect and it has added the 3rd range into the innerlist as oppose to the 4th.

** Process exited - Return Code: 0 **

TEST CASE 2: {1,3}, {2,8}, {6,10}, {9,14} = FAIL

THIS IS THE OUTER LIST: [[[1, 10], [6, 10]]]

I am going to write quickly why I have chosen to modify the counter to 1
I am now also beginning to understand that there will be stability in this exercise if there are three ranges given initially.
But when it gets beyond this, certain conditions in code is triggering different sections of the code... Which is not always fit for purpose.

I am now beginning to realize that processing this in a real time manner whilst retrieving the Objects is proving to be extremely difficult. I can foresee that if the entire Objects were stored into another repository it might have been easier.. But this is not a certainty...

The good news is that the code is no longer crashing....
It is a case of controlling the flow.

I need another design this time with paper driven approach.