NOTE: TO SIMPLIFY TESTING AND AVOIDING CHANGING VARIABLES AND FINDING IT DIFFICULT TO IDENTIFY FAULTS, I HAVE CREATED THE FOLLOWING AREAS OF THE CODE.... ANYONE WHO REQUIRES UAT TESTING CAN MODIFY THESE AREAS.....

1) These are my failed numerals during the testing phase

```
50          String []failingNumeralToDecimal = new String[]
51          {
52  /*444 Decimal: 444 => */ "CDXLIV",
53  /*449 Decimal: 449 => */"CDXLIX",
54  /*494 Decimal: 494 =>*/ "CDXCIV",
55  /*499 Decimal: 499 => */"CDXCIX",
56  /*944 Decimal: 944 => */"CMXLIV",
57  /*949 Decimal: 949 => */"CMXLIX",
58  /*994 Decimal: 994 => */"CMXCIV",
59  /*999 Decimal: 999 => */"CMXCIX",
60  /*1444 Decimal: 1444 => */"MCDXLIV",
```

2) These are all the valid numerals to be tested. With exception of the first one, this simulates a rogue numeral... It will just ensure code is capable of both scenarios:

ALSO NOTE, I HAVE USED LOGIC THROUGHOUT THE WHOLE PROCESS. THERE IS NOT ANYTHING IN MY CODE WITH HARD LOGIC MAPPING ROMAN NUMERAL => DECIMAL. THESE ARE JUST ARRAY LOCATIONS TO STORE VALUES AS PART OF THE AUTOMATED TESTING....

```
85  String [] allNumerals = new String[4000];
86
87
88  //allNumerals[0]="IVXV";   //This can be enabled to see outcome invalid numeral
89  allNumerals[0]="I";
90  allNumerals[1]="II";
91  allNumerals[2]="III";
92  allNumerals[3]="IV";
93  allNumerals[4]="V";
94  allNumerals[5]="VI";
95  allNumerals[6]="VII";
96  allNumerals[7]="VIII";
97  allNumerals[8]="IX";
98  allNumerals[9]="X";
99  allNumerals[10]="XI";
100 allNumerals[11]="XII";
101 allNumerals[12]="XIII";
102 allNumerals[13]="XIV";
```

This goes all the way up to. NOTE: I am terminating at 3999 due to irregularities after

4000 in Roman numerals is represented as IV with a bar or vinculum above it, written as $\overline{IV}$.

```
4082  allNumerals[3993]="MMMCMXCIV";
4083  allNumerals[3994]="MMMCMXCV";
4084  allNumerals[3995]="MMMCMXCVI";
4085  allNumerals[3996]="MMMCMXCVII";
4086  allNumerals[3997]="MMMCMXCVIII";
4087  allNumerals[3998]="MMMCMXCIX";
```

3) I have set up areas like this so I can test necessary array above and the range within easily.. Note it is useful if the session is killed due to memory usage of screen ouput... I found it better to do this, than to reduce the screen outputs since.....

```
4106      //for (int z=0; z<500; z++)                                  //SEVERAL OPTION TO SPEED UP TEST CASES
4107      //for (int z=1943; z<1944; z++)                              //SEVERAL OPTION TO SPEED UP TEST CASES
4108      //for (int z=1943; z<1945; z++)                              //SEVERAL OPTION TO SPEED UP TEST CASES
4109      for (int z=0; z<allNumerals.length; z++)
4110      //for (int z=0; z<failingNumeralToDecimal.length; z++)        //SEVERAL OPTION TO SPEED UP TEST CASES
```

4) area where end user can select....

```
4116          number=allNumerals[z].length();                        //SEVERAL OPTION TO SPEED UP TEST CASES
4117          //number=failingNumeralToDecimal[z].length();          //SEVERAL OPTION TO SPEED UP TEST CASES
```

5) Final area where end user can select

```
4131      do
4132      {
4133          //reader = new Scanner(System.in); // Reading from System.in
4134          //System.out.println("Enter roman numeral " + (count+1) + " of " + number + ":"); // end user prompted to enter roman numeral
4135
4136
4137          inputtedNumerals[count]=allNumerals[z].charAt(count);              //SEVERAL OPTION TO SPEED UP TEST CASES
4138          //inputtedNumerals[count]=failingNumeralToDecimal[z].charAt(count);     //SEVERAL OPTION TO SPEED UP TEST CASES
```

JUST ENSURE THAT REFERENCES ARE ALL EITHER TO ALLNUMERALS OR FAILINGNUMERALS OTHERWISE IT WILL GIVE DEVASTATING OUTCOME

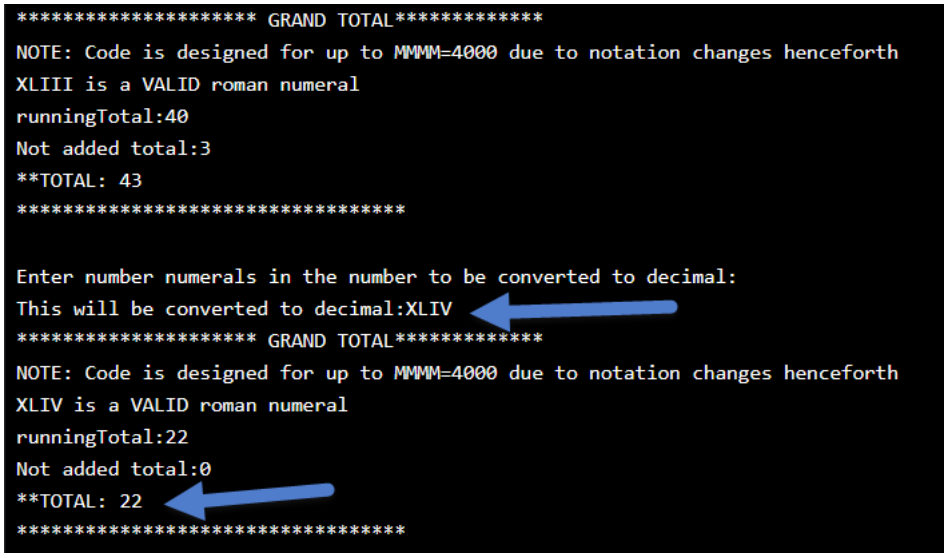
TEST CASE 1: My main objective was just to tailor my code so that it runs all numerals disable the debugging = FAIL

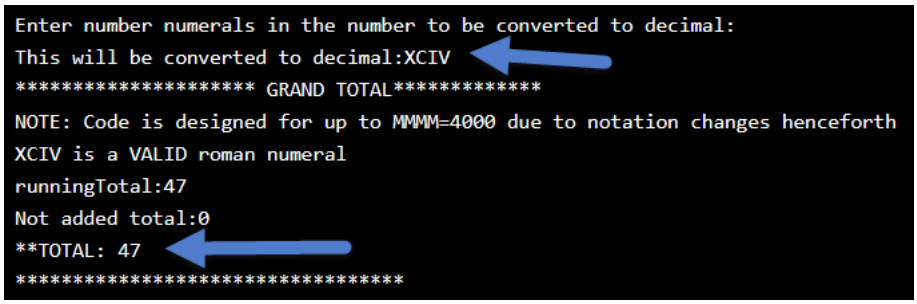
```
******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
MMMCMXCIX is a VALID roman numeral
runningTotal:99
Not added total:4000
**TOTAL: 4099
********************************

Enter number numerals in the number to be converted to decimal:
Exception in thread "main" java.lang.NullPointerException
   at Main.main(Main.java:4065)
```

At this moment, this is my lowest priority since it has failed at the end of the code.

TEST CASE 2: Checking each entry individually to ascertain if it is the correct conversion = FAIL

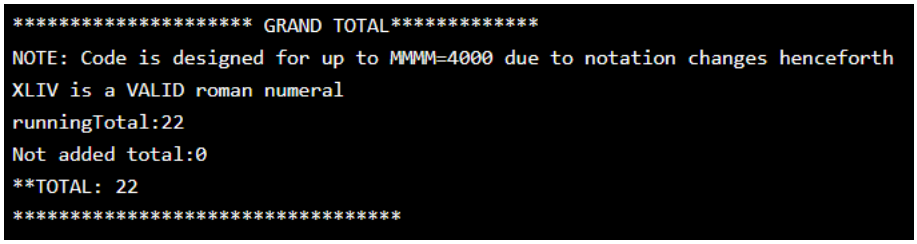
This is the first time I am seeing my software (written in 2024) fail. There are first two examples and I can see many others in my outputs! All this is reminiscent of having to divide certain totals by 2 under circumstances.





I will quickly try this numeral in the original code. I suspect issue will persist.

TEST CASE 2a; Testing numeral XLIV = FAIL against: https://www.amitamlani.com/26032024/1/Main.java (This was code extensively tested)



This was of little surprise seeing that I evolved my automated code from this.

TEST CASE 2b; Testing numeral XLIV = PASS against: https://www.amitamlani.com/26032024/2/Main.java

```
--- Note: It relies on old logic calculations from rule4Single:(0)splitting runningTotal under circumstances
******************** NEW LOGIC:  GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
XLIV is a VALID roman numeral
runningTotal:44
Not added total:0
**TOTAL: 44
********************************
```

This was the code I managed to locate on my machine before my trip to Cancun on 18 March 2025. I had tested it tentatively ONLY..
So now, my focus will be to upgrade this code to automated version.
I believe it is a case of copying the main method across and also removing the old programme logic output (at end of code). I will maintain all debugging information even though it prevents execution to 4000.

TEST CASE 2c; Testing numeral XLIV = PASS XCIV=PASS against:
https://www.amitamlani.com/26032024/2/Main.java (previously tentatively tested)
This code has executed to its limit...
It reached 509, but I can be seen there are errors that have occurred in this range.

TEST CASE 3: Commenting this section of code from = FAIL
https://www.amitamlani.com/26032024/2/Main.java (previous all passed in 2024) and will see the outcome.

```
if (!illegalSubtractive && !IplacedIncorrect && !rule3Fail && !illegalFourInARowState)
{
    //if (runningTotal>0 && unTotal==0 && FlagSet==true)
    //{
//        runningTotal = runningTotal / 2;
 //   }
```

I have commented this section, so that it looks similar to the other code....

```
******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation chang
CCCXCIX is a VALID roman numeral
runningTotal:99
Not added total:300
**TOTAL: 399
*********************************

Enter number numerals in the number to be converted to decimal:
This will be converted to decimal:CD
******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation chang
CD is a VALID roman numeral
runningTotal:500
Not added total:0
**TOTAL: 500
*********************************
```

I can see it is jumbling a lot of the numerals expected to be 4XX into 5XX.
This seems much more erroneous than having the above code present.

It appears that both versions are failing at certain points.
To keep this manageable, I will export results for these versions (with and without this section of code). It seems all errors are between 400 – 500, 3400-3500....
I am storing the master main method at bottom of this document.

<u>TEST CASE 4: Create a spreadsheet showing systematic testing to identify number range(s) causing issues for both Java classes</u>

I ascertained there were less failed conversions in this version:
https://www.amitamlani.com/26032024/2/Main.java
So I decided to remediate this version..

<u>TEST CASE 5: Homing into certain sections and devising if conditions to ensure the roman numeral tallies up..</u>
https://www.amitamlani.com/26032024/2/temp/Main.java

So I have reduced significantly the number of failed conversions... (from approximately over 400 (not this was more of a heatmap taking into several anomalies with ranges), all the way down to 25.
Now I can see that it is not a case like previously in which it alternated in a range between pass an fail.... This has made my troubleshooting much easier now since I can

focus on different variables that were returned in my old logic.

I have introduced these two loops by closely examining the old logic outputs of variables....

```
    if (rule4Single<runningTotal && rule4Single!=0 && runningTotal!=0)
    {
System.out.println("THIS RULE****************************************1");
        runningTotal=rule4Single;
    }


    if (rule6==rule7 && rule6!=0 && rule7!=0  && unTotal!=0 && unTotal%2==0)
    {
System.out.println("THIS RULE****************************************2");
        System.out.println("UNTOTAL IS: " + unTotal);
        unTotal = (unTotal/2) + rule6;
    }
```

I need to be extremely careful that any further logic I implement to resolve failed conversions such as: 400, 445, 449, 494, 499, 944, 949, 1099, 1445, 1449, 1494, 1499, 2444, 2499, 2944, 2949, 2994, 2999, 3445, 3450, 3495, 3500, 3994

I can see almost all of these are represented with 4 and 9... This is where subtractive notations are enforced... The most odd anomaly is 400.. Since this is exactly CD.

<u>TEST CASE 6: Identifying the issue with incorrect roman numeral for 400</u>
This became my first priority. Not only because it was the first failed case. But it was one of the most elementary huma error and it had not flagged up until now.
I believe the error debugging assisted me heavily...

```
4285
4286        int [][] subtractiveNumeralsRange = new int[6][2];
4287
4288        subtractiveNumeralsRange[0][0] = 1;   subtractiveNumeralsRange[0][1] = 4;
4289        subtractiveNumeralsRange[1][0] = 1;   subtractiveNumeralsRange[1][1] = 9;
4290        subtractiveNumeralsRange[2][0] = 10;   subtractiveNumeralsRange[2][1] = 40;
4291        subtractiveNumeralsRange[3][0] = 10;   subtractiveNumeralsRange[3][1] = 90;
4292        subtractiveNumeralsRange[4][0] = 100;   subtractiveNumeralsRange[4][1] = 400;
4293        subtractiveNumeralsRange[5][0] = 100;   subtractiveNumeralsRange[5][1] = 900;
4294
```

I had this value set as 500. It should have represented end of the subtractive range for (CD), which is 400... I corrected this immediately.

TEST CASE 7: Checking the conversion again for CD = PASS

TEST CASE 8: Running through all the numerals again, created another column on spreadsheet AF

These are the numbers that are failing. It can be seen that all of them have failed as a result of this newly created if loop

Unfortunately I can see that although it fixed several issues, there are numbers which previously passed that are now failing.

We know that since none of the rule values are being changed or unTotal is not being made into 0, entering one statement can not influence decision of another...

```
1052 Decimal: 1046 => MXLVI          allNumerals[1045]="MXLVI";
```

This passed with bottom two methods but failed when top one was introduced

```
    if (rule2==0 && rule6<unTotal && rule6!=0 && unTotal!=0)
    {
        System.out.println("THIS RULE*************************************3");
        unTotal=rule6;
    }
    if (rule4Single<runningTotal && rule4Single!=0 && runningTotal!=0)
    {
System.out.println("THIS RULE*************************************1");
        runningTotal=rule4Single;
    }

    if (rule6==rule7 && rule6!=0 && rule7!=0  && unTotal!=0 && unTotal%2==0)
    {
System.out.println("THIS RULE*************************************2");
        System.out.println("UNTOTAL IS: " + unTotal);
        unTotal = (unTotal/2) + rule6;
    }
```

1046 passed with these two methods but failed when the top one was introduced

********** NEW LOGIC: GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MXLVI is a VALID roman numeral

rule4single: 40

runningtotal:40

runningTotal:40

Not added total:1006

**TOTAL: 1046

****

****

OLD PROGRAMME LOGIC

****

MXLVI is the roman numeral

number of Vs: 1

number of Ls: 1

number of Ds: 0

rule4: 0

rule2: 0

rule4Single: 40 //We can try to create a condition based on this statement

rule5: 6 //We can try to create a condition based on this statement

rule6: 40 //We can try to create a condition based on this statement

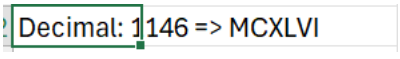
rule7: 0

threeinarow:0

Untotal:1006

We can see that rule4single is equal to rule6single
I can perhaps try to check another case to see if this is the same test case outcome...

This also passed with two if rules, so I will check if there is similarity to above...



It also has:

rule4Single: 40

rule5: 6

rule6: 40

rule7: 0

I have now narrowed down the criteria further as below and re-instaed this.

```
if (rule2==0 && rule6<unTotal && rule6!=0 && unTotal!=0 && rule4Single!=rule6)
{
    System.out.println("THIS RULE*****************************************3");
    unTotal=rule6;
}
```

TEST CASE: Testing numeral 1146 and 1046 and any other numerals which passed without using the above condition, then failed above (excluding red part) and now testing with entire block above = PASS
This proves that code is remediative.

TEST CASE: column AK Same as test case above, but now going through entire ranges that failed under similar circumstsances...

It now passes

```
******************** NEW LOGIC:  GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
MCXCVI is a VALID roman numeral
rule4single: 90
runningtotal:90
runningTotal:90
Not added total:1106
**TOTAL: 1196
```

My main interest is to try block range such as 14XX in which issues started to mount
It has removed all major errors and left me with issue on;
Decimal: 1444 => MCDXLIV (this has been problem with and without the if rule3 condition....
Decimal: 1446 => MCDXLVI (this has been problem only with if rule 3)..

So I will quickly examine 1446, similar before without this condition.
And see why it entered here and place measured to prevent it from entering...

****

OLD PROGRAMME LOGIC

****

MCDXLVI is the roman numeral

number of Vs: 1

number of Ls: 1

number of Ds: 1

rule4: 0

rule2: 0

rule4Single: 0 performing rule4Single!=rule6 will force this into if 3a, but we want to keep it out of the loop

```
if (rule2==0 && rule6<unTotal && rule6!=0 && unTotal!=0 && rule4Single!=rule6)
{
    System.out.println("THIS RULE*****************************************3");
    unTotal=rule6;
}
```

We can see there is absolutely no difference between variables populated in this numeral compared to one below... The only visible difference is that rule4Single =0 (here) and on MXLVI, rule4Single !=0
So I can increase the logic even further in this if loop

```
if (rule2==0 && rule6<unTotal && rule6!=0 && unTotal!=0 && rule4Single!=rule6  && rule4Single!=0)
{
    System.out.println("THIS RULE*****************************************3");
    unTotal=rule6;
}
```

rule5: 6

rule6: 400

rule7: 0

threeinarow:0

Untotal:1006

MCDXLVI is NOT a VALID roman numeral

Address the issues outputted for a valid Roman numeral

****

MXLVI is the roman numeral

number of Vs: 1

number of Ls: 1

number of Ds: 0

rule4: 0

rule2: 0

rule4Single: 40 //We can try to create a condition based on this statement

rule5: 6 //We can try to create a condition based on this statement

rule6: 40 //We can try to create a condition based on this statement

rule7: 0

threeinarow:0

Untotal:1006

TEST CASE: I will now try the more enhanced conditional loop and run through some of the numerals that failed in Column AF. I will record now in column AK...

Decimal: 146 => CXLVI = PASS
Decimal: 196 => CXCVI = PASS
Decimal: 445 => CDXLV = FAIL but this never failed when if rule 3 was more simple
Decimal: 449 => CDXLIX = FAIL but this never failed when if rule 3 was more simple

So it has failed to enter this loop. Since rule4single is 0, it has not entered here....

```
if (rule2==0 && rule6<unTotal && rule6!=0 && unTotal!=0 && rule4Single!=rule6  && rule4Single!=0)
{
    System.out.println("THIS RULE*****************************************3");
    unTotal=rule6;
}
```

CDXLIV is the roman numeral

number of Vs: 1

number of Ls: 1

number of Ds: 1

rule4: 0

rule2: 0

rule4Single: 0

rule5: 0

rule6: 400

rule7: 0

threeinarow:0

Untotal:500

But it has passed on this :
(t enters since it does not check rule4single)...

```
if (rule2==0 && rule6<unTotal && rule6!=0 && unTotal!=0 && rule4Single!=rule6)
{
    System.out.println("THIS RULE*****************************************3");
    unTotal=rule6;
}
```

CDXLIV is the roman numeral

number of Vs: 1

number of Ls: 1

number of Ds: 1

rule4: 0

rule2: 0

rule4Single: 0

rule5: 0

rule6: 400

rule7: 0

threeinarow:0

Untotal:500

This has left the testing extremely difficult since there is no other conditions I can try to bring in apart from relaxing the if statement..
If I do not find a solution, I will just it enter the if loop (i.e create a repeat loop) without rule4Single!=0 And only let decimals in certain range into it (I will keep track of those numbers).

Whilst the above problems are issues, I am trying those which have not been fixed by any of my new rule conditional statements..

TEST CASE: = PASS

Decimal: 2944 => MMCMXLIV
It has been outputting 3044 initially and now 2044... I have had to change my logic. However this code had massive effect elsewhere os it was removed.

```java
if (rule6==rule7 && rule5!=0 && !inside)
{
    System.out.println("THIS RULE*****************************************4");
    unTotal = rule6 + rule2;
    inside=true;
    //runningTotal =
}
```

```
--- Note: It relies on old logic calculations from rule4Single:(0)s
******************** NEW LOGIC:  GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes
MMCMXLIV is a VALID roman numeral
rule4single: 0
runningtotal:44
THIS RULE*****************************************4
runningTotal:44
Not added total:2900
**TOTAL: 2944
**********************************
```

```
*****************************************
OLD PROGRAMME LOGIC

*****************************************
MMCMXLIV is the roman numeral
number of Vs: 1
number of Ls: 1
number of Ds: 0
rule4: 0
rule2: 2000
rule4Single: 0
rule5: 1100
rule6: 900
rule7: 900
threeinarow:0
Untotal:2900
```

I am now going to run all the numerals again..
And I will just need to accept the outcome that arrives.....
I have definitely tried everything possible in my means.

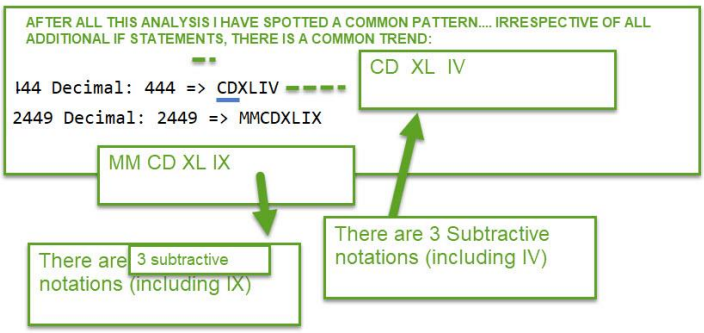
<u>FAILED NUMBERS</u>
444, 449, 494, 499, 944.949, 994,999, 1444,1449,1494,1499,1944,1949,1994,1999,
2444,2449,2494,2499, 2949,2994,2999,3444,3449,3494,3499,3944,3949,3999

I have now just taken opportunity to run through my original code and also implement the three if statements. To make this easier, my results are recorded at the top. I will also place its roman numeral next to it, which will be useful to identify pattern.
Depending on the variations, it might require a different approach to resolve this...
But it can clearly be seen a pattern at the top numbers..
I feel now I have to look beyond the rules and the patterns in the numbers above that have emerged....

TEST CASE: Run both my original codes and removed any unnecessary rules which I had no logical explanation for

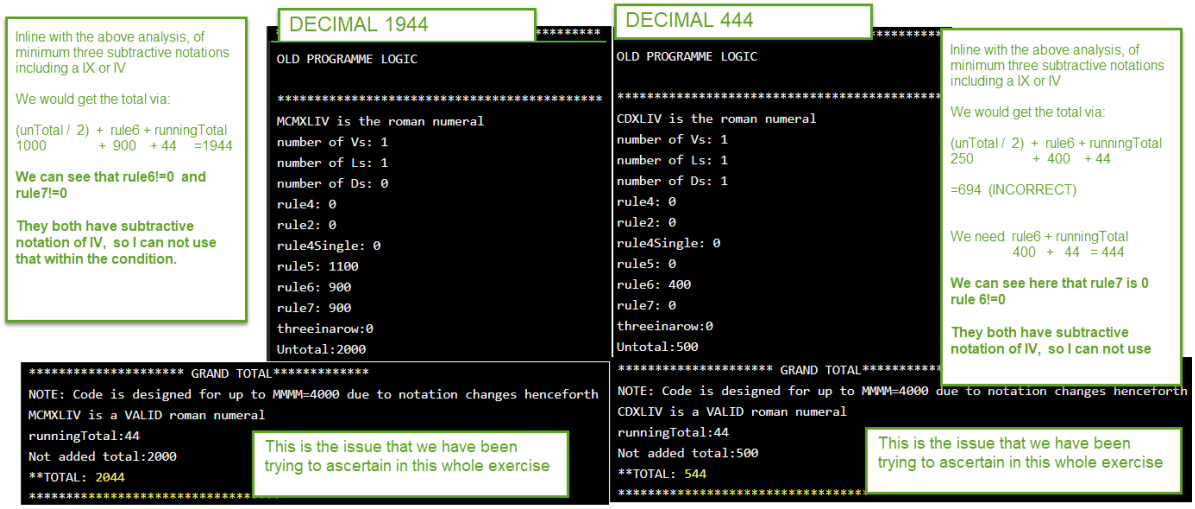
I have identified a pattern... I am now sharing the information as images on the site...

```
444 Decimal: 444 => CDXLIV
449 Decimal: 449 => CDXLIX
494 Decimal: 494 => CDXCIV
499 Decimal: 499 => CDXCIX
944 Decimal: 944 => CMXLIV
949 Decimal: 949 => CMXLIX
994 Decimal: 994 => CMXCIV
999 Decimal: 999 => CMXCIX
1444 Decimal: 1444 => MCDXLIV
1449 Decimal: 1449 => MCDXLIX
1494 Decimal: 1494 => MCDXCIV
1499 Decimal: 1499 => MCDXCIX
1944 Decimal: 1944 => MCMXLIV
1999 Decimal: 1999 => MCMXCIX
2444 Decimal: 2444 => MMCDXLIV
2449 Decimal: 2449 => MMCDXLIX
2494 Decimal: 2494 => MMCDXCIV
2499 Decimal: 2499 => MMCDXCIX
2944 Decimal: 2944 => MMCMXLIV
2949 Decimal: 2949 => MMCMXLIX
2994 Decimal: 2994 => MMCMXCIV
2999 Decimal: 2999 => MMCMXCIX
3444 Decimal: 3444 => MMMCDXLIV
3449 Decimal: 3449 => MMMCDXLIX
3494 Decimal: 3494 => MMMCDXCIV
3499 Decimal: 3499 => MMMCDXCIX
3944 Decimal: 3944 => MMMCMXLIV
3949 Decimal: 3949 => MMMCMXLIX
3999 Decimal: 3999 => MMMCMXCIX
```



I can conclude that there has to be a minimum of three subtractive notations (and one of them is IV or IX).

I will apply this logic into my automated versions of the code... and get rid of any of the extra two or three if statements as part of exercise above. It was bizarre that all codes with or without these produced same failed numerals. But this was not the case during first phase of testing!



The logic is even simpler, I can deduce that if it has three subtractivenotations (initial numeral), it is guaranteed to have and IV or IX in it...

TEST CASE: Implement logic to check for three subtractive notations including IV or IX within entire numeral.
Also create the if condition to derive an alternate total

I have added this area on new code:

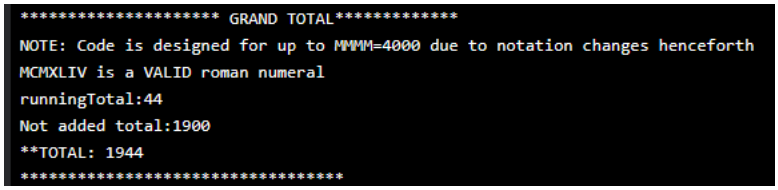
```
for (int v=0; v<storeSubtractiveNumerals.length;v++)
{
    //need a nested for loop
    if (storeSubtractiveNumerals[v]==subtractiveNumerals[j])
    {
        isStoredJvalue = true;

    }
    if (storeSubtractiveNumerals[v]==subtractiveNumerals[i])
    {
        isStoredIvalue = true;
    }
}
```

```
if (!isStoredIvalue)
    {
    storeSubtractiveNumerals[indexLocation]=subtractiveNumerals[i];
    indexLocation++;
    }

    if (!isStoredJvalue)
    {
    storeSubtractiveNumerals[indexLocation]=subtractiveNumerals[j];
    indexLocation++;
    }
    //we now check if it has three
    if (indexLocation==2)
    {
        isFullstoreThreeSubtractiveNumerals=true;
    }
```

TEST CASE: Check Decimal: 444 => CDXLIV = PASS

```
******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
CDXLIV is a VALID roman numeral
runningTotal:44
Not added total:400
**TOTAL: 444
*********************************
```

TEST CASE: Check Decimal: 1944 => MCMXLIV = PASS



I will now go through all the failing test cases… It is best I put them in a separate array…
And see how the code executes

TEST CASE: Create a String array for all failing numerals and execute the code = PASS
This is the first time in all my coding efforts have these numerals passed.

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXLIV is a VALID roman numeral

runningTotal:44

Not added total:400

**TOTAL: 444

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXLIX is a VALID roman numeral

runningTotal:49

Not added total:400

**TOTAL: 449

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXCIV is a VALID roman numeral

runningTotal:94

Not added total:400

**TOTAL: 494

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXCIX is a VALID roman numeral

runningTotal:99

Not added total:400

**TOTAL: 499

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXLIV is a VALID roman numeral

runningTotal:44

Not added total:900

**TOTAL: 944

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXLIX is a VALID roman numeral

runningTotal:49

Not added total:900

**TOTAL: 949

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXCIV is a VALID roman numeral

runningTotal:94

Not added total:900

**TOTAL: 994

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXCIX is a VALID roman numeral

runningTotal:99

Not added total:900

**TOTAL: 999

****

********** GRAND TOTAL********** /FAIL should be 1444

**** difficult with available values in rules variables

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXLIV is a VALID roman numeral

runningTotal:44

Not added total:1150

**TOTAL: 1194

****

********** GRAND TOTAL********** /FAIL should be 1449

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXLIX is a VALID roman numeral

runningTotal:49

Not added total:1150

**TOTAL: 1199

****

********** GRAND TOTAL********** /FAIL should be 1494

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXCIV is a VALID roman numeral

runningTotal:94

Not added total:1150

**TOTAL: 1244

****

********** GRAND TOTAL********** /FAIL should be 1499

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXCIX is a VALID roman numeral

runningTotal:99

Not added total:1150

**TOTAL: 1249

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCMXLIV is a VALID roman numeral

runningTotal:44

Not added total:1900

**TOTAL: 1944

****

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCMXCIX is a VALID roman numeral

runningTotal:99

Not added total:1900

**TOTAL: 1999

****

********** GRAND TOTAL********** should be 2444

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXLIV is a VALID roman numeral

runningTotal:44

Not added total:1650

**TOTAL: 1694

****

********** GRAND TOTAL********** should be 2449

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXLIX is a VALID roman numeral

runningTotal:49

Not added total:1650

**TOTAL: 1699

****

********** GRAND TOTAL********** should be 2494

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXCIV is a VALID roman numeral

runningTotal:94

Not added total:1650

**TOTAL: 1744

****

********** GRAND TOTAL********** should be 2499

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXCIX is a VALID roman numeral

runningTotal:99

Not added total:1650

**TOTAL: 1749

****

********** GRAND TOTAL********** should be 2944

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCMXLIV is a VALID roman numeral

runningTotal:44

Not added total:2400

**TOTAL: 2444

****

********** GRAND TOTAL********** should be 2949

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

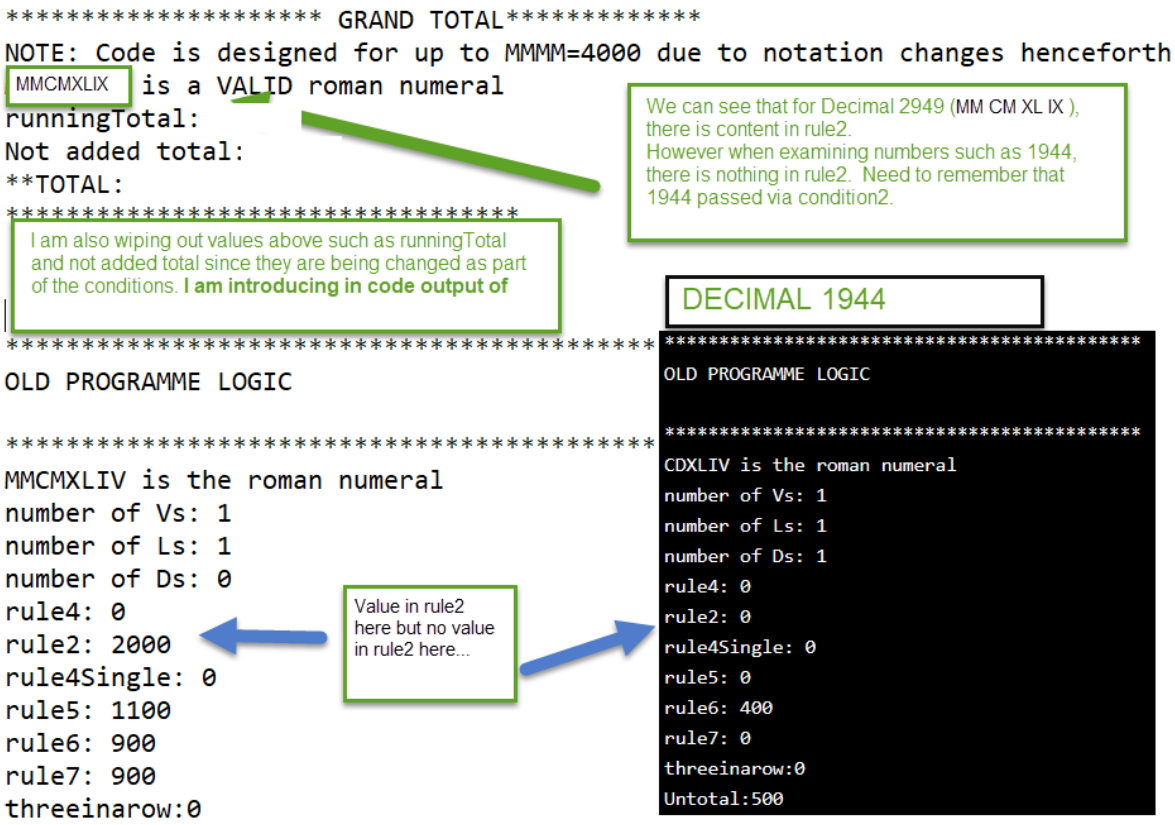
MMCMXLIX is a VALID roman numeral

runningTotal:49

Not added total:2400

**TOTAL: 2449

****



So I have had to modify the logic in all my Conditions

```java
if (isFullstoreThreeSubtractiveNumerals)
{
    if (rule6!=0 && rule7!=0 && rule2==0)
    {
        unTotal = (unTotal/2) + rule6;
        System.out.println("THIS CONDITION1");
    }
    //test case for decimal 444 or 1944
    if (rule6!=0 && rule7==0 && rule2==0)
    {
        unTotal = rule6;
        System.out.println("THIS CONDITION2");
    }
    //test case for Decimal 2949
    if (rule6!=0 && rule7!=0 && rule2!=0)
    {
        unTotal = rule2 + rule6;   //rule7 can also work
        System.out.println("THIS CONDITION3");
    }
```

TEST CASE: I will need to run through entire set of numerals again. And ensure there are no disruptions…

Decimal: 444 => CDXLIV

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXLIV is a VALID roman numeral

runningTotal:44

Not added total:400

**TOTAL: 444

****

Decimal: 449 => CDXLIX

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXLIX is a VALID roman numeral

runningTotal:49

Not added total:400

**TOTAL: 449

****

Decimal: 494 => CDXCIV

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXCIV is a VALID roman numeral

runningTotal:94

Not added total:400

**TOTAL: 494

****

Decimal: 499 => CDXCIX

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CDXCIX is a VALID roman numeral

runningTotal:99

Not added total:400

**TOTAL: 499

****

Decimal: 944 => CMXLIV

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXLIV is a VALID roman numeral

runningTotal:44

Not added total:900

**TOTAL: 944

****

Decimal: 949 => CMXLIX

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXLIX is a VALID roman numeral

runningTotal:49

Not added total:900

**TOTAL: 949

****

Decimal: 994 => CMXCIV

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXCIV is a VALID roman numeral

runningTotal:94

Not added total:900

**TOTAL: 994

****

Decimal: 999 => CMXCIX

THIS CONDITION2

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

CMXCIX is a VALID roman numeral

runningTotal:99

Not added total:900

**TOTAL: 999

****

ERROR WITH THIS.. WE CAN SEE IT HAS GONE INTO CONDITION 1 now and total has been affected
Decimal: 1444 => MCDXLIV

THIS CONDITION1

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXLIV is a VALID roman numeral

runningTotal:44

Not added total:1150

**TOTAL: 1194

ERROR WITH THIS
Decimal: 1449 => MCDXLIX

THIS CONDITION1

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXLIX is a VALID roman numeral

runningTotal:49

Not added total:1150

**TOTAL: 1199

Decimal: 1449 => MCDXLIX

Here is the result that I was getting (see below) and the intervention required.

```
**********************************ORIGINAL UNTOTAL: 1500
THIS CONDITION1


******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
MCDXLIX is a VALID roman numeral
runningTotal:49
Not added total:1150
**TOTAL: 1199
**********************************
```

(rule5 - 100) + rule6 + runningTotal

1000 + 400 + 49 = **1449**

```
*****************************************
OLD PROGRAMME LOGIC

*****************************************
MCDXLIX is the roman numeral
number of Vs: 0
number of Ls: 1
number of Ds: 1
rule4: 0
rule2: 0
rule4Single: 0
rule5: 1100
rule6: 400
rule7: 400
threeinarow:0
Untotal:1150
```

The logic holds so I will adjust the if statement for condition1

FROM:

```
if (rule6!=0 && rule7!=0 && rule2==0)
{
    unTotal = (unTotal/2) + rule6;
    System.out.println("THIS CONDITION1");
}
```

TO:

```
if (rule6!=0 && rule7!=0 && rule2==0)
{
    unTotal = (rule5-100) + rule6;
    System.out.println("THIS CONDITION1");
}
```

```
**********************************
**********************************ORIGINAL UNTOTAL: 1500
THIS CONDITION1


******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
MCDXLIV is a VALID roman numeral
runningTotal:44
Not added total:1150
**TOTAL: 1194
**********************************


******************************************
OLD PROGRAMME LOGIC

******************************************
MCDXLIV is the roman numeral
number of Vs: 1
number of Ls: 1
number of Ds: 1
rule4: 0
rule2: 0
rule4Single: 0
rule5: 1100
rule6: 400
rule7: 400
threeinarow:0
Untotal:1150
Enter number numerals in the number to be converted to decimal:
This will be converted to decimal:MCDXLIX
the length is:7
```

Only way to make target 1,444 is performing

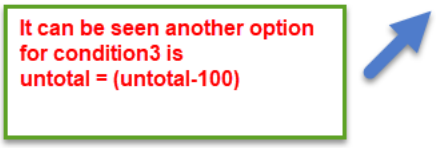
(rule5-100) + (rule 6 or rule 7) + runningTotal

1000 + 400 + 44 = **1444**

Perhaps this has arisen due to the CD subtractive notation

If this logic holds for the remaining failed cases below, I will need to consider it a valid solution

```
**********************************ORIGINAL UNTOTAL: 1500
THIS CONDITION1
WHAT IS RUNNING TOTAL: 44

******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation ch
MCDXLIV is a VALID roman numeral
runningTotal:44
Not added total:1400
**TOTAL: 1444
**********                ***********

*****************************************
OLD PROGRAMME LOGIC

*****************************************
MCDXLIV is the roman numeral
number of Vs: 1
number of Ls: 1
number of Ds: 1
rule4: 0
rule2: 0
rule4Single: 0
rule5: 1100
rule6: 400
rule7: 400
threeinarow:0
Untotal:1400
```



I will try all the cases again and notably look out for when it changes to condition1

Decimal: 1444 => MCDXLIV

THIS CONDITION1

WHAT IS RUNNING TOTAL: 44

WHAT IS untotal: 1400

********** GRAND TOTAL*************

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXLIV is a VALID roman numeral

runningTotal:44

Not added total:1400

**TOTAL: 1444

****

Decimal: 1449 => MCDXLIX

THIS CONDITION1

WHAT IS RUNNING TOTAL: 49

WHAT IS untotal: 1400

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXLIX is a VALID roman numeral

runningTotal:49

Not added total:1400

**TOTAL: 1449

****

Decimal: 1494 => MCDXCIV

THIS CONDITION1

WHAT IS RUNNING TOTAL: 94

WHAT IS untotal: 1400

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXCIV is a VALID roman numeral

runningTotal:94

Not added total:1400

**TOTAL: 1494

Decimal: 1499 => MCDXCIX

THIS CONDITION1

WHAT IS RUNNING TOTAL: 99

WHAT IS untotal: 1400

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCDXCIX is a VALID roman numeral

runningTotal:99

Not added total:1400

**TOTAL: 1499

****

Decimal: 1944 => MCMXLIV

THIS CONDITION1

WHAT IS RUNNING TOTAL: 44

WHAT IS untotal: 1900

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCMXLIV is a VALID roman numeral

runningTotal:44

Not added total:1900

**TOTAL: 1944

****

Decimal: 1999 => MCMXCIX

THIS CONDITION1

WHAT IS RUNNING TOTAL: 99

WHAT IS untotal: 1900

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MCMXCIX is a VALID roman numeral

runningTotal:99

Not added total:1900

**TOTAL: 1999

****

Decimal: 2444 => MMCDXLIV

THIS CONDITION3 //We are now in condition 3 and it still seems to be ok

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXLIV is a VALID roman numeral

runningTotal:44

Not added total:2400

**TOTAL: 2444

****

Decimal: 2449 => MMCDXLIX

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXLIX is a VALID roman numeral

runningTotal:49

Not added total:2400

**TOTAL: 2449

****

Decimal: 2494 => MMCDXCIV

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXCIV is a VALID roman numeral

runningTotal:94

Not added total:2400

**TOTAL: 2494

****

Decimal: 2499 => MMCDXCIX

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCDXCIX is a VALID roman numeral

runningTotal:99

Not added total:2400

**TOTAL: 2499

****

Decimal: 2944 => MMCMXLIV

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCMXLIV is a VALID roman numeral

runningTotal:44

Not added total:2900

**TOTAL: 2944

****

Decimal: 2949 => MMCMXLIX

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCMXLIX is a VALID roman numeral

runningTotal:49

Not added total:2900

**TOTAL: 2949

****

Decimal: 2994 => MMCMXCIV

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCMXCIV is a VALID roman numeral

runningTotal:94

Not added total:2900

**TOTAL: 2994

****

Decimal: 2999 => MMCMXCIX

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMCMXCIX is a VALID roman numeral

runningTotal:99

Not added total:2900

**TOTAL: 2999

****

Decimal: 3444 => MMMCDXLIV

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCDXLIV is a VALID roman numeral

runningTotal:44

Not added total:3400

**TOTAL: 3444

****

Decimal: 3449 => MMMCDXLIX
THIS CONDITION3
********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCDXLIX is a VALID roman numeral

runningTotal:49

Not added total:3400

**TOTAL: 3449

****

Decimal: 3494 => MMMCDXCIV
THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCDXCIV is a VALID roman numeral

runningTotal:94

Not added total:3400

**TOTAL: 3494

****

Decimal: 3499 => MMMCDXCIX
THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCDXCIX is a VALID roman numeral

runningTotal:99

Not added total:3400

**TOTAL: 3499

****

Decimal: 3944 => MMMCMXLIV

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCMXLIV is a VALID roman numeral

runningTotal:44

Not added total:3900

**TOTAL: 3944

****

Decimal: 3949 => MMMCMXLIX

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCMXLIX is a VALID roman numeral

runningTotal:49

Not added total:3900

**TOTAL: 3949

****

Decimal: 3999 => MMMCMXCIX

THIS CONDITION3

********** GRAND TOTAL**********

NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth

MMMCMXCIX is a VALID roman numeral

runningTotal:99

Not added total:3900

**TOTAL: 3999

****

I can safely say I have seen every conversion appear on the screen successfully. But with all the changing, I need to run through entire numerals again....

I am hoping this will be my final test case..

TEST CASE: Running through entire execution... = PASS

For some reason the code is not terminating... and re-enters the for loop.

```
******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
MMMCMXCIX is a VALID roman numeral
runningTotal:99
Not added total:3900
**TOTAL: 3999
********************************



*****************************************
OLD PROGRAMME LOGIC

*****************************************
MMMCMXCIX is the roman numeral
number of Vs: 0
number of Ls: 0
number of Ds: 0
rule4: 0
rule2: 3000
rule4Single: 0
rule5: 1100
rule6: 900
rule7: 900
threeinarow:0
Untotal:3900
Enter number numerals in the number to be converted to decimal:
Exception in thread "main"
java.lang.NullPointerException
        at Main.main(Main.java:4114)
```

I have created a

TEST CASE: Trying an array with invalid roman numeral and a valid after PASS

Roman Numeral V can not precede: 10 //These are type messages I want to capture in errorLog and present it at end after the new Logic to ensure end user has better visibility

Illegal subtractive notation found with V, D or L //suitable for error log

Rule 7

This is overall total:0

This is value of c: 1

This is the numeral value: 1

This is value of c: 1

This is the numeral value: 5

This is value of c: 0

This is the numeral value: 10

This has not been added to the total10

This is the position: 2

This is value of c: 0

This is the numeral value: 5

This has not been added to the total15

This is the position: 3

Invalid roman numeral. Numeral V or D or L has occured more than once //suitable for error log

********** **GRAND TOTAL**********

IVXV is an INVALID roman numeral

Address the issues outputted for a valid Roman numeral

****

****

OLD PROGRAMME LOGIC

****

IVXV is the roman numeral

number of Vs: 2

number of Ls: 0

number of Ds: 0

rule4: 0

rule2: 0

rule4Single: 4

rule5: 0

rule6: 4

rule7: 0

threeinarow:0

Untotal:15

More than 2 instances of V, L and D: //suitable for error log

D: 0 //suitable for error log

V: 2 //suitable for error log

L: 0 //suitable for error log

IVXV is NOT a VALID roman numeral

Address the issues outputted for a valid Roman numeral

TEST CASE: Tidying up error log messages. Then inputting rogue numeral (input with no knowledge of the system) and following logs to change it to valid numeral

My strategy is to remove the first error reported and then re-compile code until it passes as valid....

I had to increase this variable to compensate for more subtractive notations

```
//if the roman numeral was valid this array size =3 would suffice
String []storeSubtractiveNumerals = new String[(numeralsToString.length()/2)];
```

OUTPUT *******

Welcome to Online IDE!! Happy Coding :)

Enter number numerals in the number to be converted to decimal:

This will be converted to decimal:IVXCXIVCILCDCCMMIVCMXXXXXICDLLM

the length is:31

numerals in the roman numeral:31

In rule 4

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

IV 0

total after rule4:0

true

****

IX -1

total after rule4:0

true

****

XL -1

total after rule4:0

true

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

XC

This should print out start of the range:10

This should print out end of the range:90

IV

This should print out start of the range:1

This should print out end of the range:4

correct order

XC 2

IV 0

TWO SUBTRACTIVE NOTATIONS FOUND:

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

XC 2

total after rule4:0

true

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

CD

This should print out start of the range:100

This should print out end of the range:400

IV

This should print out start of the range:1

This should print out end of the range:4

TWO SUBTRACTIVE NOTATIONS FOUND:

****

CD

This should print out start of the range:100

This should print out end of the range:400

XC

This should print out start of the range:10

This should print out end of the range:90

TWO SUBTRACTIVE NOTATIONS FOUND:

****

CD 10

total after rule4:0

true

****

TWO SUBTRACTIVE NOTATIONS FOUND:

****

CM

This should print out start of the range:100

This should print out end of the range:900

IV

This should print out start of the range:1

This should print out end of the range:4

TWO SUBTRACTIVE NOTATIONS FOUND:

****

CM

This should print out start of the range:100

This should print out end of the range:900

XC

This should print out start of the range:10

This should print out end of the range:90

TWO SUBTRACTIVE NOTATIONS FOUND:

****

CM

This should print out start of the range:100

This should print out end of the range:900

CD

This should print out start of the range:100

This should print out end of the range:400

CM is in the range: 100 - 900

CD is in the range: 100 - 400

They are both in the same range

REACH HERE!!!!!!!!!!

CM 13

total after rule4:0

true

Going into rule 3

value of indexCount: 0

31

value of indexCount: 1

31

found a V

value of indexCount: 2

31

value of indexCount: 3

31

value of indexCount: 4

31

value of indexCount: 5

31

value of indexCount: 6

31

found a V

value of indexCount: 7

31

value of indexCount: 8

31

value of indexCount: 9

31

value of indexCount: 10

31

value of indexCount: 11

31

value of indexCount: 12

31

value of indexCount: 13

31

value of indexCount: 14

31

value of indexCount: 15

31

value of indexCount: 16

31

value of indexCount: 17

31

found a V

value of indexCount: 18

31

value of indexCount: 19

31

value of indexCount: 20

31

value of indexCount: 21

31

value of indexCount: 22

31

value of indexCount: 23

31

value of indexCount: 24

31

value of indexCount: 25

31

value of indexCount: 26

31

value of indexCount: 27

31

value of indexCount: 28

31

value of indexCount: 29

31

value of indexCount: 30

31

What is value of zeroindexcount: 0

How many times does it enter in this section

What is m: 10

What is m-1: 100

at rule 6: 0

Rule 6

Must WORK!!!!

0

1

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 1

How many times does it enter in this section

What is m: 1

What is m-1: 10

*******RULE5******: 11

at rule 6: 0

Rule 6

Must WORK!!!!

1

5

Roman Numeral V(5) (numeral index:1) can not precede: X(10) **//I will remove the V from the numeral**

Illegal subtractive notation found with V, D or L

Rule 7

total at rule 7:4

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 2

at rule 6: 0

Rule 6

Must WORK!!!!

2

10

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 3

at rule 6: 0

Rule 6

Must WORK!!!!

3

100

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 4

How many times does it enter in this section

What is m: 1

What is m-1: 100

*******RULE5******: 101

at rule 6: 0

Rule 6

Must WORK!!!!

4

10

Rule 7

total at rule 7:53

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 5

at rule 6: 0

Rule 6

Must WORK!!!!

5

1

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 6

How many times does it enter in this section

What is m: 100

What is m-1: 500

at rule 6: 0

Rule 6

Must WORK!!!!

6

5

Roman Numeral V(5) (numeral index:6) can not precede: C(100) **//I will remove the V from the numeral**

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

starting rule 1

Value of coutner: 0

Value of coutner: 4

Value of coutner: 8

Value of coutner: 12

Value of coutner: 16

Value of coutner: 20

Value of coutner: 24

Value of coutner: 28

Amit Amlani

What is value of zeroindexcount: 7

at rule 6: 0

Rule 6

Must WORK!!!!

7

100

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 8

How many times does it enter in this section

What is m: 1

What is m-1: 1000

*******RULE5******: 1001

at rule 6: 0

Rule 6

Must WORK!!!!

8

1

Roman Numeral I(1) (numeral index:8) can not precede: L(50) **//I will remove the I from the numeral**

Illegal subtractive notation found with V, D or L

Rule 7

total at rule 7:57

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 9

at rule 6: 0

Rule 6

Must WORK!!!!

9

50

Roman Numeral L(50) (numeral index:9) can not precede: C(100) **//I will remove the L from the numeral**

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 10

at rule 6: 0

Rule 6

Must WORK!!!!

10

100

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 11

at rule 6: 0

Rule 6

Must WORK!!!!

11

500

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 12

How many times does it enter in this section

What is m: 10

What is m-1: 1000

*******RULE5******: 1010

at rule 6: 0

Rule 6

Must WORK!!!!

12

100

Rule 7

This is overall total:0

Going into rule 3

starting rule 1

Value of coutner: 0

Value of coutner: 4

Value of coutner: 8

Value of coutner: 12

Value of coutner: 16

Value of coutner: 20

Value of coutner: 24

Value of coutner: 28

Amit Amlani

So far sum:20

does this happen45

What is value of zeroindexcount: 13

at rule 6: 0

Rule 6

Must WORK!!!!

13

100

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

starting rule 1

Value of coutner: 0

Value of coutner: 4

Value of coutner: 8

Value of coutner: 12

Value of coutner: 16

Value of coutner: 20

Value of coutner: 24

Value of coutner: 28

Amit Amlani

So far sum:30

does this happen45

What is value of zeroindexcount: 14

at rule 6: 0

Rule 6

Must WORK!!!!

14

1000

Rule 7

This is overall total:0

Going into rule 3

starting rule 1

Value of coutner: 0

Value of coutner: 4

Value of coutner: 8

Value of coutner: 12

Value of coutner: 16

Value of coutner: 20

Value of coutner: 24

Value of coutner: 28

Amit Amlani

So far sum:40

does this happen45

What is value of zeroindexcount: 15

at rule 6: 0

Rule 6

Must WORK!!!!

15

1000

Rule 7

This is overall total:0

Going into rule 3

starting rule 1

Value of coutner: 0

Value of coutner: 4

Value of coutner: 8

Value of coutner: 12

Value of coutner: 16

Value of coutner: 20

Value of coutner: 24

Value of coutner: 28

Amit Amlani

So far sum:50

What is value of zeroindexcount: 16

at rule 6: 0

Rule 6

Must WORK!!!!

16

1

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 17

How many times does it enter in this section

What is m: 1

What is m-1: 10

*******RULE5******: 11

at rule 6: 0

Rule 6

Must WORK!!!!

17

5

Roman Numeral V(5) (numeral index:17) can not precede: C(100) //I will remove the V from the numeral

Illegal subtractive notation found with V, D or L

Rule 7

total at rule 7:156

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 18

at rule 6: 0

Rule 6

Must WORK!!!!

18

100

Illegal subtractive notation found with V, D or L

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 19

at rule 6: 0

Rule 6

Must WORK!!!!

19

1000

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 20

How many times does it enter in this section

What is m: 50

What is m-1: 500

*******RULE5******: 550

at rule 6: 0

Rule 6

Must WORK!!!!

20

10

Rule 7

This is overall total:0

Going into rule 3

starting rule 1

Amit Amlani

So far sum:150

What is value of zeroindexcount: 21

at rule 6: 0

Rule 6

Must WORK!!!!

21

10

Rule 7

This is overall total:0

Going into rule 3

What is value of zeroindexcount: 22

at rule 6: 0

Rule 6

Must WORK!!!!

22

10

Rule 7

This is overall total:0

This is value of c: 1

This is the numeral value: 1

This is value of c: 1

This is the numeral value: 5

This is value of c: 1

This is the numeral value: 10

This is value of c: 1

This is the numeral value: 100

This is value of c: 0

This is the numeral value: 10

This has not been added to the total: 10

This is the position: 4

This is value of c: 0

This is the numeral value: 1

This has not been added to the total: 11

This is the position: 5

This is value of c: 0

This is the numeral value: 5

This has not been added to the total: 16

This is the position: 6

This is value of c: 0

This is the numeral value: 100

This has not been added to the total: 116

This is the position: 7

This is value of c: 0

This is the numeral value: 1

This has not been added to the total: 117

This is the position: 8

This is value of c: 0

This is the numeral value: 50

This has not been added to the total: 167

This is the position: 9

This is value of c: 1

This is the numeral value: 100

This is value of c: 1

This is the numeral value: 500

This is value of c: 0

This is the numeral value: 100

This has not been added to the total: 267

This is the position: 12

This is value of c: 1

This is the numeral value: 100

This is value of c: 1

This is the numeral value: 1000

This is value of c: 0

This is the numeral value: 1000

This has not been added to the total: 1267

This is the position: 15

This is value of c: 0

This is the numeral value: 1

This has not been added to the total: 1268

This is the position: 16

This is value of c: 0

This is the numeral value: 5

This has not been added to the total: 1273

This is the position: 17

This is value of c: 0

This is the numeral value: 100

This has not been added to the total: 1373

This is the position: 18

This is value of c: 0

This is the numeral value: 1000

This has not been added to the total: 2373

This is the position: 19

This is value of c: 0

This is the numeral value: 10

This has not been added to the total: 2383

This is the position: 20

This is value of c: 0

This is the numeral value: 10

This has not been added to the total: 2393

This is the position: 21

This is value of c: 0

This is the numeral value: 10

This has not been added to the total: 2403

This is the position: 22

This is value of c: 0

This is the numeral value: 10

This has not been added to the total: 2413

This is the position: 23

This is value of c: 0

This is the numeral value: 10

This has not been added to the total: 2423

This is the position: 24

This is value of c: 0

This is the numeral value: 1

This has not been added to the total: 2424

This is the position: 25

This is value of c: 0

This is the numeral value: 100

This has not been added to the total: 2524

This is the position: 26

This is value of c: 0

This is the numeral value: 500

This has not been added to the total: 3024

This is the position: 27

This is value of c: 0

This is the numeral value: 50

This has not been added to the total: 3074

This is the position: 28

This is value of c: 0

This is the numeral value: 50

This has not been added to the total: 3124

This is the position: 29

This is value of c: 0

This is the numeral value: 1000

This has not been added to the total: 4124

This is the position: 30

Invalid roman numeral. Numeral V or D or L has occured more than once

********** GRAND TOTAL**********

IVXCXIVCILCDCCMMIVCMXXXXXICDLLM is an INVALID roman numeral

Address the issues outputted for a valid Roman numeral

****

****

OLD PROGRAMME LOGIC

****

IVXCXIVCILCDCCMMIVCMXXXXXICDLLM is the roman numeral

number of Vs: 3

number of Ls: 3

number of Ds: 2

rule4: 0

rule2: 150

rule4Single: 0

rule5: 550

rule6: 4

rule7: 156

threeinarow:0

Untotal:4124

More than 2 instances of V, L and D:

D: 2

V: 3

L: 3

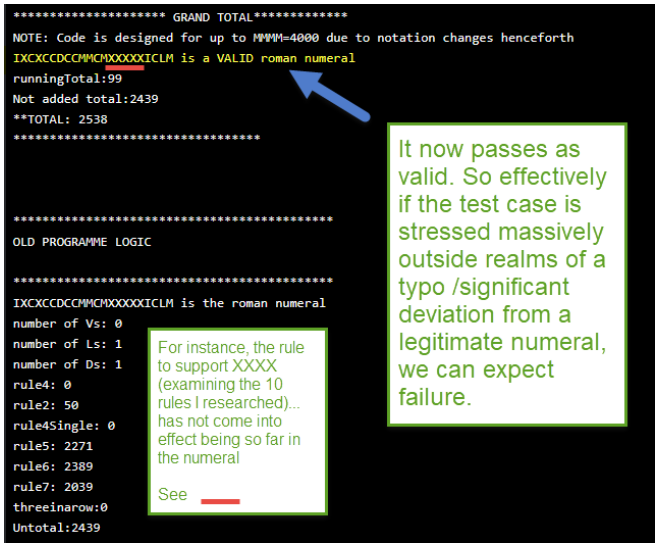
IVXCXIVCILCDCCMMIVCMXXXXXICDLLM is NOT a VALID roman numeral

Address the issues outputted for a valid Roman numeral

I eventually finish here and no more error messages . But it prompts of the repeat D and L.
So I will remove the repeat that occurs

```
*****************************************
OLD PROGRAMME LOGIC

*****************************************
IXCXCCDCCMMCMXXXXXICDLLM is the roman numeral
number of Vs: 0
number of Ls: 2
number of Ds: 2
rule4: 0
rule2: 150
rule4Single: 0
rule5: 2671
rule6: 2389
rule7: 1089
threeinarow:0
Untotal:3951
More than 2 instances of V, L and D:
D: 2
V: 0
L: 2
IXCXCCDCCMMCMXXXXXICDLLM is NOT a VALID roman numeral
Address the issues outputted for a valid Roman numeral
```

Once I finish this, my numeral has validated as being legitimate. It is clearly not. So it shows that my code can not be stressed outside realms of a small typo inputted by the end user.



THIS IS AN EXAMPLE OF A SITUATION THAT I HAD EARLY ON AND THIS WAS DUE TO ME NOT SETTING THE INDEX VALUE CORRECTLY... FORTUNATELY THIS WAS STRAIGHT FORWARD
BUT IT SHOWS THAT THIS CHALLENGE WAS ON A COMPLETE KNIFE EDGE

```
THIS CONDITION2


********** GRAND TOTAL*********
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
CDXIV is a VALID roman numeral
runningTotal:404
Not added total:400
**TOTAL: 804
**********


*****************************************
OLD PROGRAMME LOGIC

*****************************************
CDXIV is the roman numeral
number of Vs: 1
number of Ls: 0
number of Ds: 1
rule4: 0
rule2: 0
rule4Single: 0
rule5: 0
rule6: 400
rule7: 0
threeinarow:0
Untotal:400
Enter number numerals in the number to be converted to decimal:
This will be converted to decimal:CDXV
the length is:4
numerals in the roman numeral:4
In rule 4
```

This was expected to be 414 but it has reached this loop and failed.... This was not a failing case before.....

```java
//I believe if there are three subtractive notations, it has to contain a IV or IX, so I am not checking this
if (isFullstoreThreeSubtractiveNumerals)
{
    if (rule6!=0 && rule7!=0)
    {
        unTotal = (unTotal/2) + rule6;
        System.out.println("THIS CONDITION1");
    }

    if (rule6!=0 && rule7==0)
    {
        unTotal = rule6;
        System.out.println("THIS CONDITION2");
    }
}
```

It will be interesting to see which numerals are stored in here since the numeral CDXIV has two subtractive notations ONLY

```
This is overall total:0
This is value of c: 1
This is the numeral value: 100
This is value of c: 1
This is the numeral value: 500
This is value of c: 0
This is the numeral value: 10
This has not been added to the total10
This is the position: 2
This is value of c: 1
This is the numeral value: 1
This is value of c: 1
This is the numeral value: 5
THE NUMERAL: IV
THE NUMERAL: CD
THE NUMERAL: null
THIS CONDITION2
```

This was incorrectly set to 2

```java
if (indexLocation==3)
{
    isFullstoreThreeSubtractiveNumerals=true;
}
```

Fortunately it was a beginners mistake. I had the value above set to to 2 as oppose to 3....

Hence it set the boolean and triggered below even though there are two subtractive notations

```java
//I believe if there are three subtractive not
if (isFullstoreThreeSubtractiveNumerals)
{
```

```
******************** GRAND TOTAL*************
NOTE: Code is designed for up to MMMM=4000 due to notation changes henceforth
CDXIV is a VALID roman numeral
runningTotal:404
Not added total:400
**TOTAL: 804
************************************
```