```
Note: Permutation.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

Welcome to Online IDE!! Happy Coding :)

***PERMUTATIONS***

P(n,r) = n! / (n−r)!

P(9,9) = 9! / (9-9)!

There are : 362880 permutations of arranging  3 x 3 grid

There are : 108,883,584,818,776,183,656,945,007,213,012,309,135,068,193,536,000
permutations of arranging  3 x 3 grid into 9 x 9:P(362880,9)

There are : 6,670,903,752,021,072,936,960 permutations of completing sudoku (taken
from internet)

This code will attempt to explore but its impossible to expect much

It is used for foundation of experimentation but also it has made serious attempt to
complete random process to make a grid

I am removing excess code so it is ready future development.

JUST ONCE            //I created a condition here so that it performs this once

This has been chosen: 5,6,4,8,9,7,2,3,1 unique: 0  //these are all unique entries

This has been chosen: 1,2,3,4,5,6,7,8,9 unique: 1  //these are all unique entries

This has been chosen: 4,5,6,7,8,9,1,2,3 unique: 2  //these are all unique entries

This has been chosen: 6,4,5,9,7,8,3,1,2 unique: 3  //these are all unique entries

This has been chosen: 2,3,1,5,6,4,8,9,7 unique: 4  //these are all unique entries

This has been chosen: 7,8,9,1,2,3,4,5,6 unique: 5  //these are all unique entries

This has been chosen: 8,9,7,2,3,1,5,6,4 unique: 6  //these are all unique entries

This has been chosen: 3,1,2,6,4,5,9,7,8 unique: 7  //these are all unique entries

This has been chosen: 9,7,8,3,1,2,6,4,5 unique: 8  //these are all unique entries

FINISHED GETTING ALL THE MINI 3x3 GRIDS  //I have kept this once it has broken out of
the do while loop where it has obtained all unique entries

String converted: 9  //it can be seen that it getting Unique 8.. This is the first time I
have seen code is not going to expectations
```

```
String converted: 7

String converted: 8

String converted: 3

String converted: 1

String converted: 2

String converted: 6

String converted: 4

String converted: 5
```

//it can be seen that it getting Unique 8.. This is cumulative issue from the last bit of logic

```
PERFORMED REAL TIME: 9(0,0)


PERFORMED REAL TIME: 9(0,0) 7(0,1)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1)


PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2)
```

//It has successfully finished populating the 3x3 mini grid

//I wouldn’t expect it to perform the number generation again since it has only performed real time filling once and it has already got all mini 3 x 3 gridsl.
I have to remember that it previously relied on an entryset and increased this by one in order to get next grid.
This has been chosen: 3,1,2,6,4,5,9,7,8 unique: 0 //these are all unique entries, we can also see that it is a different selection to previously

This has been chosen: 7,8,9,1,2,3,4,5,6 unique: 1

This has been chosen: 8,9,7,2,3,1,5,6,4 unique: 2

This has been chosen: 1,2,3,4,5,6,7,8,9 unique: 3

This has been chosen: 4,5,6,7,8,9,1,2,3 unique: 4

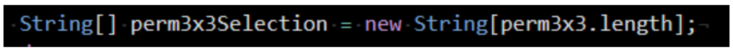
This has been chosen: 5,6,4,8,9,7,2,3,1 unique: 5

This has been chosen: 2,3,1,5,6,4,8,9,7 unique: 6

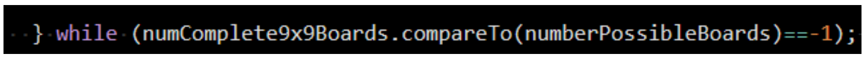
This has been chosen: 6,4,5,9,7,8,3,1,2 unique: 7

This has been chosen: 9,7,8,3,1,2,6,4,5 unique: 8

IT MUST BE HERE NOW
//same pattern appearing again where it is processing last entry... But we can see that coincidentally that this is also the same grid that has already been populated since this random number has been selected at last uniqueEntry
This is because it is hitting the end of the while loop and then reaching the top again....
And re-initialising the variable...
So effectively.....
WE NEED TO ENSURE THAT EACH OF THE CHOSEN ENTRIES ABOVE ARE PASSED INTO REALTIME9X9 AS OPPOSE TO LOOPING TO THE TOP AGAIN....



```
 String[] perm3x3Selection = new String[perm3x3.length];
```



```
  } while (numComplete9x9Boards.compareTo(numberPossibleBoards)==-1);
```

String converted: 9

String converted: 7

String converted: 8

String converted: 3

String converted: 1

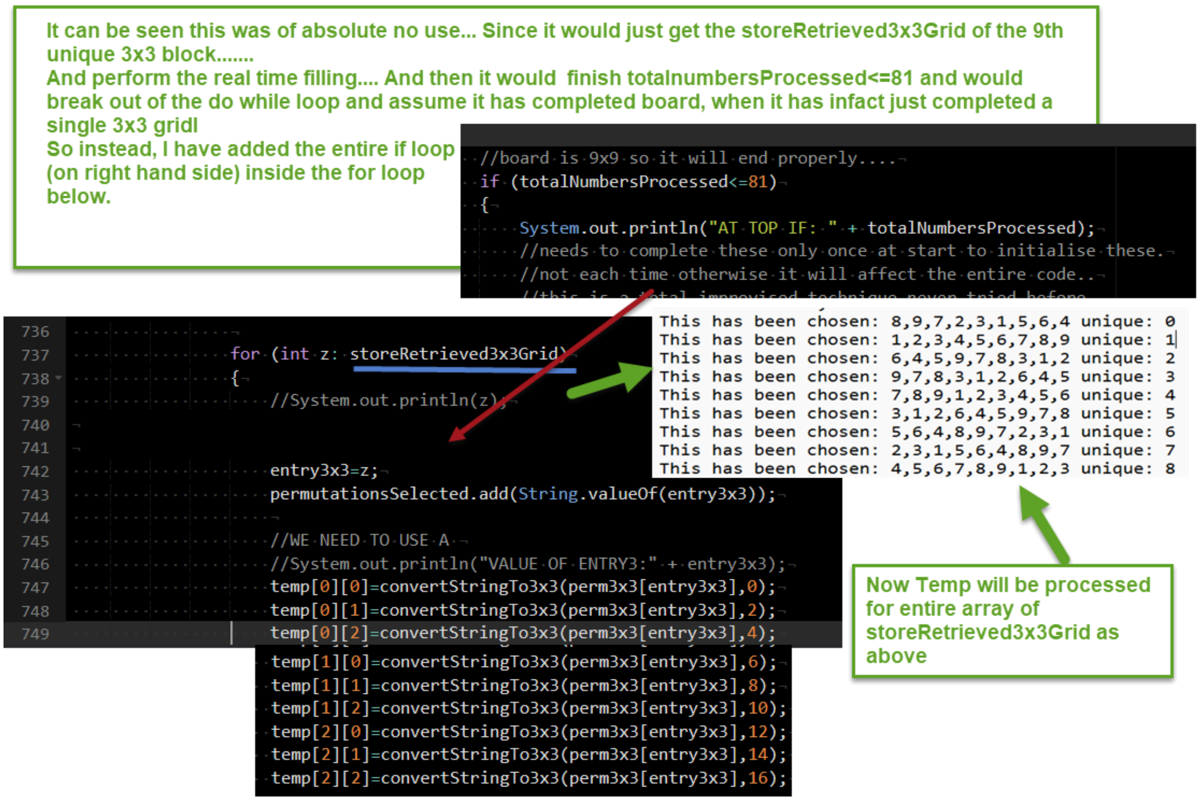
String converted: 2

String converted: 6

String converted: 4

```
String converted: 5

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5) 3(1,3)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5) 3(1,3) 1(1,4)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5) 3(1,3) 1(1,4) 2(1,5)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5) 3(1,3) 1(1,4) 2(1,5) 6(2,3)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5) 3(1,3) 1(1,4) 2(1,5) 6(2,3) 4(2,4)

PERFORMED REAL TIME: 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2) 9(0,3)
7(0,4) 8(0,5) 3(1,3) 1(1,4) 2(1,5) 6(2,3) 4(2,4) 5(2,5)
```

I added lots of screen outputs and found several logical errors. My instinct suggests it is related to usage of entrySet3x3 variable.


It can be seen this was of absolute no use... Since it would just get the storeRetrieved3x3Grid of the 9th unique 3x3 block.......
And perform the real time filling.... And then it would finish totalnumbersProcessed<=81 and would break out of the do while loop and assume it has completed board, when it has infact just completed a single 3x3 gridl
So instead, I have added the entire if loop (on right hand side) inside the for loop below.

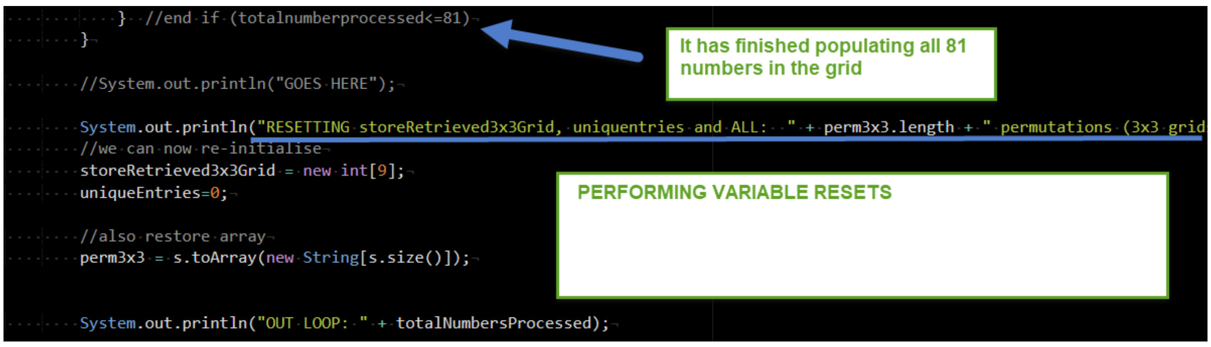
```java
//board is 9x9 so it will end properly....
if (totalNumbersProcessed<=81)
{
    System.out.println("AT TOP IF: " + totalNumbersProcessed);
    //needs to complete these only once at start to initialise these.
    //not each time otherwise it will affect the entire code..
    //this is a total improvised technique never tried before
```

```java
                for (int z: storeRetrieved3x3Grid)
                {
                    //System.out.println(z);


                    entry3x3=z;
                    permutationsSelected.add(String.valueOf(entry3x3));

                    //WE NEED TO USE A
                    //System.out.println("VALUE OF ENTRY3:" + entry3x3);
                    temp[0][0]=convertStringTo3x3(perm3x3[entry3x3],0);
                    temp[0][1]=convertStringTo3x3(perm3x3[entry3x3],2);
                    temp[0][2]=convertStringTo3x3(perm3x3[entry3x3],4);
                    temp[1][0]=convertStringTo3x3(perm3x3[entry3x3],6);
                    temp[1][1]=convertStringTo3x3(perm3x3[entry3x3],8);
                    temp[1][2]=convertStringTo3x3(perm3x3[entry3x3],10);
                    temp[2][0]=convertStringTo3x3(perm3x3[entry3x3],12);
                    temp[2][1]=convertStringTo3x3(perm3x3[entry3x3],14);
                    temp[2][2]=convertStringTo3x3(perm3x3[entry3x3],16);
```

```
This has been chosen: 8,9,7,2,3,1,5,6,4 unique: 0
This has been chosen: 1,2,3,4,5,6,7,8,9 unique: 1
This has been chosen: 6,4,5,9,7,8,3,1,2 unique: 2
This has been chosen: 9,7,8,3,1,2,6,4,5 unique: 3
This has been chosen: 7,8,9,1,2,3,4,5,6 unique: 4
This has been chosen: 3,1,2,6,4,5,9,7,8 unique: 5
This has been chosen: 5,6,4,8,9,7,2,3,1 unique: 6
This has been chosen: 2,3,1,5,6,4,8,9,7 unique: 7
This has been chosen: 4,5,6,7,8,9,1,2,3 unique: 8
```

Now Temp will be processed for entire array of storeRetrieved3x3Grid as above

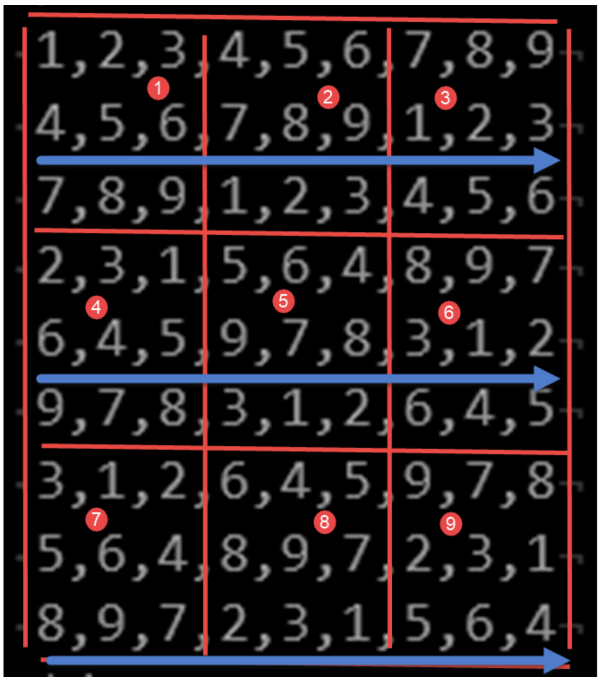
I also would complete the tidy up and reset of variables once it performed the full board.


```java
            }  //end if (totalnumberprocessed<=81)
        }

    //System.out.println("GOES HERE");

    System.out.println("RESETTING storeRetrieved3x3Grid, uniquentries and ALL:  " + perm3x3.length + " permutations (3x3 grid
    //we can now re-initialise
    storeRetrieved3x3Grid = new int[9];
    uniqueEntries=0;

    //also restore array
    perm3x3 = s.toArray(new String[s.size()]);


    System.out.println("OUT LOOP: " + totalNumbersProcessed);
```

It has finished populating all 81 numbers in the grid

PERFORMING VARIABLE RESETS

As an extra validation, I will display all the random numbers generated.
But we know that we need to first identify which random number corresponds to each row (since we know adding entries into Set does not maintain its natural order).



```
1,2,3,4,5,6,7,8,9
4,5,6,7,8,9,1,2,3
7,8,9,1,2,3,4,5,6
2,3,1,5,6,4,8,9,7
6,4,5,9,7,8,3,1,2
9,7,8,3,1,2,6,4,5
3,1,2,6,4,5,9,7,8
5,6,4,8,9,7,2,3,1
8,9,7,2,3,1,5,6,4
```

As a reminder, this is the ONLY way that suduko can be satisfied with 9 mini grids. They are populated as shown with blue arrows... There might be other ways, but we expect at MINIMUM for it to show completion if the following is outputted by the code

```
1 This has been chosen: 1,2,3,4,5,6,7,8,9 unique: 0
2 This has been chosen: 4,5,6,7,8,9,1,2,3 unique: 1
3 This has been chosen: 7,8,9,1,2,3,4,5,6 unique: 2
4 This has been chosen: 2,3,1,5,6,4,8,9,7 unique: 3
5 This has been chosen: 8,9,7,2,3,1,5,6,4 unique: 4
6 This has been chosen: 5,6,4,8,9,7,2,3,1 unique: 5
7 This has been chosen: 3,1,2,6,4,5,9,7,8 unique: 6
8 This has been chosen: 6,4,5,9,7,8,3,1,2 unique: 7
9 This has been chosen: 9,7,8,3,1,2,6,4,5 unique: 8
```

I also expect to be able to find correct entry in the exports as follows: (referring to uniqueEntry number)...

```
Permutations selected: ( 0,1,2,3,4,5,6,7,8 )
```

NOTE: We will need to check the values in board summary against permutation selected and expect each value above to carry same sequence

i.e 5= 9(0,0) 7(0,1) 8(0,2) 3(1,0) 1(1,1) 2(1,2) 6(2,0) 4(2,1) 5(2,2)