```
public class Sudoku {

    public String[][] makeSudoku(String s) {
        int SIZE = 9;
        int k = 0;
        String[][] x = new String[SIZE][SIZE];
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                x[i][j] = s.substring(k, k + 1);
                k++;
            }
        }
        return x;
    }

    public String getPrintableSudoku(String[][] x) {
        int SIZE = 9;
        String temp = "";
        for (int i = 0; i < SIZE; i++) {
            if ((i == 3) || (i == 6)) {
                temp = temp + "=================\n";
            }
            for (int j = 0; j < SIZE; j++) {
                if ((j == 3) || (j == 6)) {
                    temp = temp + " || ";
                }
                temp = temp + x[i][j];
            }
            temp = temp + "\n";
        }
        return temp;
    }

    public boolean isValidSudoku(String[][] x) {
        return rowsAreLatin(x) && colsAreLatin(x) && goodSubsquares(x);
    }

    public boolean rowsAreLatin(String[][] x) {
        boolean test = true;
        for (int i = 0; i < x.length; i++) {
            test = test && rowIsLatin(x,i);
        }
        return test;
    }


//    OR...Try the more efficient algorithm below:
//
//    public boolean rowsAreLatin(String[][] x)
//    {
```

```
//        boolean test = true;
//        int i = 0;
//        while (test == true && i < x.length)
//        {
//            test = test & rowIsLating(x,i);
//        }
//    }




    public boolean rowIsLatin(String[][] x, int i) {
        boolean[] soduku = new boolean[9];
      for(int j = 0; j < 9; j++){
            int number = Integer.parseInt(x[i][j]) - 1;

            if(soduku[number]){
            return false;
            }

        soduku[number] = true;
        }
        return true;
    }

    public boolean colsAreLatin(String[][] x) {
        for (int i = 0; i < x.length; i++) {
                if (!colIsLatin(x, i)) {
                    return false;
            }
        }
        return true;
    }

    public boolean colIsLatin(String[][] x, int j) {
        boolean[] soduku = new boolean[9];
      for(int k = 0; k < 9; k++)
      {
        int number = Integer.parseInt(x[k][j]) - 1;

        if(soduku[number])
        {
          return false;
        }
        soduku[number] = true;
      }
        return true;
    }

    public boolean goodSubsquares(String[][] x) {
        for (int i = 0; i < 3; i++)
        {
           for(int j = 0; j < 3; j++)
           {
             if (!goodSubsquare(x, i, j)) {
               return false;
              }
           }

        }
        return true;
    }

    public boolean goodSubsquare(String[][] x, int i, int j) {

        i = 1;
        j = 1;

        boolean[] soduku = new boolean[9];

        for(int var1 = i * 3, endOfRow = var1 + 3; var1 < endOfRow; var1++)
        {
          for(int var2 = j * 3, endOfCol = var2 + 3; var2 < endOfCol; var2++) {


            int number = Integer.parseInt(x[var1][var2]) - 1;

            if(soduku[number])
            {
                return false;
              }
              soduku[number] = true;
           }
        }
        return true;
    }

}
```