Pascal's Triangle

Given *numRows*, generate the first *numRows* of Pascal's triangle.

For example, given *numRows* = 5,
Return

```
[
     [1],
    [1,1],
   [1,2,1],
  [1,3,3,1],
 [1,4,6,4,1]
]
```

Solution 1

```
public class Solution {
public List<List<Integer>> generate(int numRows)
{
    List<List<Integer>> allrows = new ArrayList<List<Integer>>();
    ArrayList<Integer> row = new ArrayList<Integer>();
    for(int i=0;i<numRows;i++)
    {
        row.add(0, 1);
        for(int j=1;j<row.size()-1;j++)
            row.set(j, row.get(j)+row.get(j+1));
        allrows.add(new ArrayList<Integer>(row));
    }
    return allrows;

}
```

}

written by rheaxu original link here

Solution 2

two loops, one go through the row, one go through the column

database: pretty straight forward, ArrayList

calculate element value: K(i)(j)=K(i-1)(j-1)+K(i-1)(j) except for the first and last element

```
public class Solution {
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> triangle = new ArrayList<List<Integer>>();
        if (numRows <=0){
            return triangle;
        }
        for (int i=0; i<numRows; i++){
            List<Integer> row =  new ArrayList<Integer>();
            for (int j=0; j<i+1; j++){
                if (j==0 || j==i){
                    row.add(1);
                } else {
                    row.add(triangle.get(i-1).get(j-1)+triangle.get(i-1).get(j));
                }
            }
            triangle.add(row);
        }
        return triangle;
    }
}
```

written by chennan original link here

Solution 3

```
class Solution {
public:
    vector<vector<int> > generate(int numRows) {
        vector<vector<int>> r(numRows);

        for (int i = 0; i < numRows; i++) {
            r[i].resize(i + 1);
            r[i][0] = r[i][i] = 1;

            for (int j = 1; j < i; j++)
                r[i][j] = r[i - 1][j - 1] + r[i - 1][j];
        }

        return r;
    }
};
```

written by mzchen original link here

From Leetcoder.