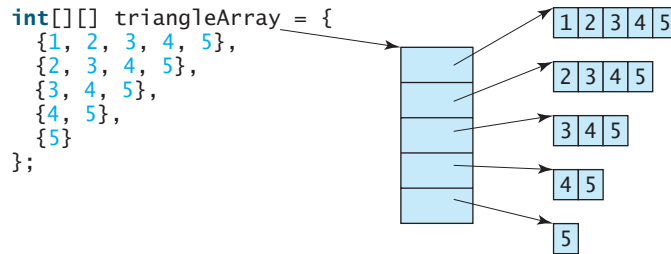


### 7.2.3 Ragged Arrays

Each row in a two-dimensional array is itself an array. Thus the rows can have different lengths. An array of this kind is known as a *ragged array*. Here is an example of creating a ragged array:



As can be seen, `triangleArray[0].length` is 5, `triangleArray[1].length` is 4, `triangleArray[2].length` is 3, `triangleArray[3].length` is 2, and `triangleArray[4].length` is 1.

If you don't know the values in a ragged array in advance, but know the sizes, say the same as before, you can create a ragged array using the syntax that follows:

```
int[][] triangleArray = new int[5][];
triangleArray[0] = new int[5];
triangleArray[1] = new int[4];
triangleArray[2] = new int[3];
triangleArray[3] = new int[2];
triangleArray[4] = new int[1];
```

You can now assign values to the array. For example,

```
triangleArray[0][3] = 50;
triangleArray[4][0] = 45;
```



#### Note

The syntax `new int[5][]` for creating an array requires the first index to be specified. The syntax `new int[][]` would be wrong.

## 7.3 Processing Two-Dimensional Arrays

Suppose an array `matrix` is created as follows:

```
int[][] matrix = new int[10][10];
```

Here are some examples of processing two-dimensional arrays:

1. (*Initializing arrays with input values*) The following loop initializes the array with user input values:

```
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
    matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        matrix[row][column] = input.nextInt();
    }
}
```

2. (*Initializing arrays with random values*) The following loop initializes the array with random values between 0 and 99:

```
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
```

```

        matrix[row][column] = (int)(Math.random() * 100);
    }
}

```

3. (*Printing arrays*) To print a two-dimensional array, you have to print each element in the array using a loop like the following:

```

for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        System.out.print(matrix[row][column] + " ");
    }

    System.out.println();
}

```

4. (*Summing all elements*) Use a variable named **total** to store the sum. Initially **total** is 0. Add each element in the array to **total** using a loop like this:

```

int total = 0;
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        total += matrix[row][column];
    }
}

```

5. (*Summing elements by column*) For each column, use a variable named **total** to store its sum. Add each element in the column to **total** using a loop like this:

```

for (int column = 0; column < matrix[0].length; column++) {
    int total = 0;
    for (int row = 0; row < matrix.length; row++)
        total += matrix[row][column];
    System.out.println("Sum for column " + column + " is " +
        total);
}

```

6. (*Which row has the largest sum?*) Use variables **maxRow** and **indexOfMaxRow** to track the largest sum and index of the row. For each row, compute its sum and update **maxRow** and **indexOfMaxRow** if the new sum is greater.

```

int maxRow = 0;
int indexOfMaxRow = 0;

// Get sum of the first row in maxRow
for (int column = 0; column < matrix[0].length; column++) {
    maxRow += matrix[0][column];
}

for (int row = 1; row < matrix.length; row++) {
    int totalOfThisRow = 0;
    for (int column = 0; column < matrix[row].length; column++)
        totalOfThisRow += matrix[row][column];

    if (totalOfThisRow > maxRow) {
        maxRow = totalOfThisRow;
        indexOfMaxRow = row;
    }
}

System.out.println("Row " + indexOfMaxRow
    + " has the maximum sum of " + maxRow);

```



**Video Note**  
find the row with the  
largest sum