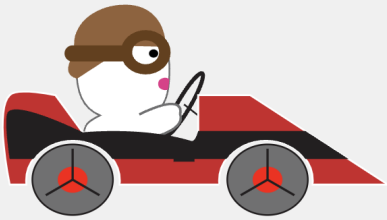
2-Dimensional Arrays



Drivers for a racing team complete test circuits.

They complete 3 test circuits every day from Monday to Friday.

The team monitors how many times a driver uses the in-car radio during each test to report a problem. This data is stored and analysed.

The data is currently stored in one-dimensional arrays as follows:

```
int[] mondayData = new int[3];

int[] tuesdayData = new int[3];

int[] wednesdayData = new int[3];

int[] thursdayData = new int[3];

int[] fridayData = new int[3];

//code to populate the arrays not shown

int count = 0;

//radoiCallCount returns the sum of all elements of an integer array

count = count + radioCallCount(mondayData);

count = count + radioCallCount(tuesdayData);

count = count + radioCallCount(wednesdayData);

count = count + radioCallCount(thursdayData);

count = count + radioCallCount(fridayData);

System.out.println("The total number of radio calls this week was "+count);
```

Task 1 - Arrays Review

Design the function radioCallCount. It processes an integer array, summing all of the elements and returning the result.

Coding this would be tricky if there were more array than 5 arrays, for example 100 or 1000!

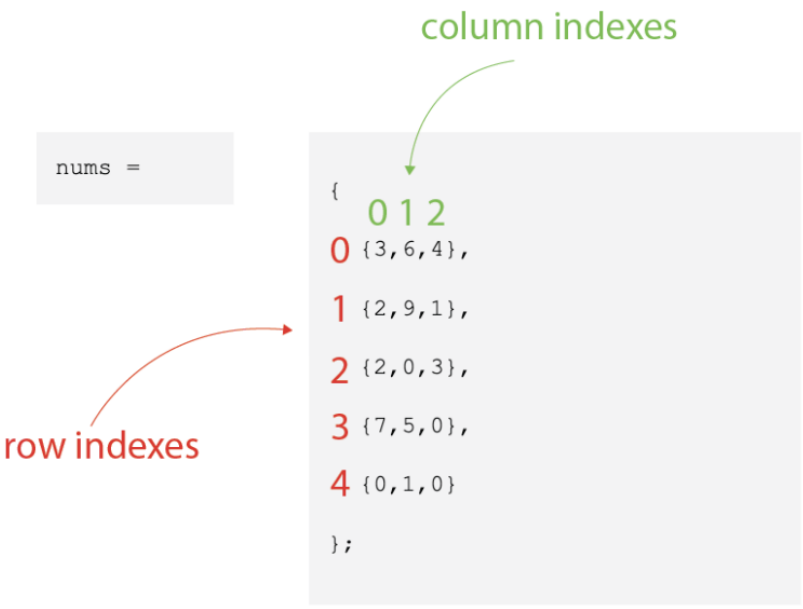
Making the call to `radioCallCount` 1000 times would involve a lot of coding!

As usual in Computer Science, programmers found a way to overcome this problem - 2-Dimensional Arrays!

A 2-Dimensional array is simply an array of arrays. For example, let's say we have 5 seperate arrays:

```
{3,6,4}

{2,9,1}

{2,0,3}

{7,5,0}

{0,1,0}
```

Let's pack them into an array and give the array an identifier!



`nums` has 5 rows and 3 columns.

The first row has an index of 0.

The first column has an index of 0.

We can see that the element at nums[1][2] has a value of 1.

So, when we declare this array of integers, we can do this in Java:

```
int[][] nums = new int[5][3];
```

If we want to process the first row, we can do this:

```
for(int i = 0; i<3; i++){
  System.out.println(nums[0][i]);
}
```

Task 2
Write some code to add up all of the elements in the 4th row.

```
public static void addFourthRow(){









}
```

In fact, to add up all the rows, we can use a nested for loop:

```
for(int i = 0; i<3; i++){
  for(int j = 0; j<3; j++){
    System.out.println(nums[i][j]);
  }
}
```

Task 3
Redesign radioCallCount so that it processes a single 2-dimensional array of 5 rows and 3 columns and returns a sum of all of the elements.

```
public static int radioCallCount(int[][] radioCalls){









}
```

If we don't know how many rows a 2D array has, we can use .length eg `nums.length`

If we don't know how many columns a row has, we can also use .length eg `nums[0].length`

In this course, we will only process arrays where every row has the same number of columns.

CODE IT #1

A function, count3s, takes a 2-dimensional integer array as a parameter. It returns a count of how many 3s are stored in the array. The function header is:

```
public static int count3s(int[][] nums)
```

Code this function and use this main program to test your design:

```
int[][] dataSet = {{3,2,2}, {1,2,3}, {1,5,6}, {4,3,3}};

int result = count3s(dataSet);

System.out.print(result);
```

CODE IT #2

A function, **`rowIndex`**, processes a 2D array. It returns the index of the row which has the greatest total. If more than 1 row has the greatest total, it will return only one of the indexes.

The function header is:

```
public static int rowIndex(int[][] nums)
```

Test your solution with this program:

```
int[][] dataSet = {{3,2,2}, {1,2,3}, {1,5,6}, {4,3,3}};

int result = rowIndex(dataSet);

System.out.print(result);
```