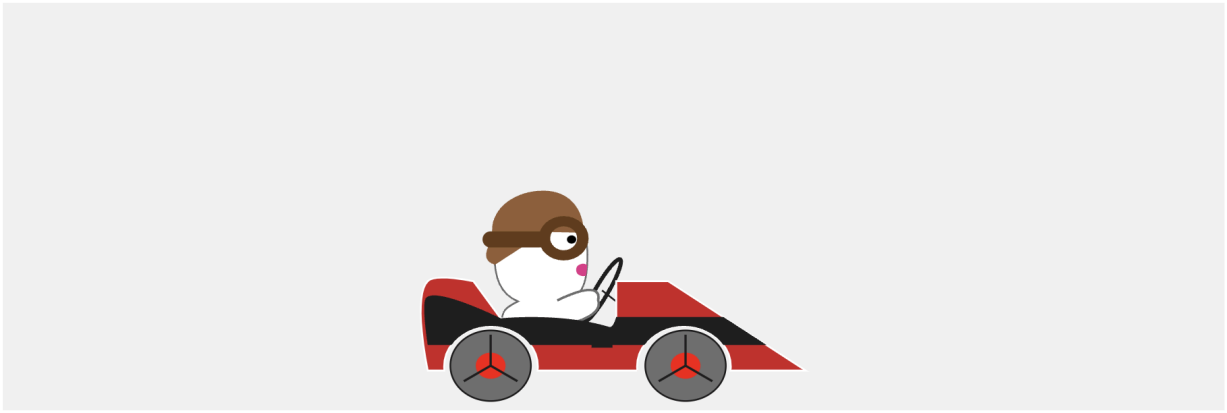
2-Dimensional Arrays



Review

When we have a lot of data that is the same data type, we can use arrays! For example, instead of this:

```
int a = 3;
int b = 7;
int c = 4;
int d = 8;
int e = -1
```

We can do this:

```
int[] nums = {3,7,4,8,-1};
```

Instead of having lots of **identifiers**[1], we only need **one identifier** to access the data. This makes processing the data easier.

Problem

But what about the following situation?

Drivers for a racing team complete test circuits.

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

[1] An identifier is just a variable name, or the name of a function

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 as follows:

int[ ] **mon** = {3,6,4};

int[ ] **tues** = {2,9,1};

int[ ] **wed** = {2,0,3};

int[ ] **thu** = {7,5,0};

int[ ] **fri** = {0,1,0};

We need 5 arrays! 5 different **identifiers**. If we need to calculate the **total** number of radio calls, it is going to be tricky!

When we have multiple arrays containing the same data type, one strategy is to create a **2-dimensional array** - an array of arrays!

```
int[][] nums = {
      {3, 6, 4},
      {2, 9, 1},
      {2, 0, 3},
      {7, 5, 0},
      {0, 1, 0}
}
```

Once populated, we could imagine it to look like this.

There are 5 **rows** and 3 **columns**.
At nums[**0**][**0**], the value 3 is stored.

Write down the values stored at the following locations in the array nums:

location	value
nums[**0**][**0**]	
nums[**3**][**2**]	
nums[**4**][**1**]	

Nested For Loop

Let's process the 2-dimensional array nums and display every value:

Java

```
for(int i = 0; i<5; i++){
    for(int j = 0; j<3; j++){
        System.out.print(nums[i][j] + " ");
    }
    System.out.println(); //next row,new line
}
```

The **outer** loop visits each row.
The **inner** loop visits each column in the current row being processed.

Coding Task 1

Implement the nested loop above inside the **`displayRadioCalls`** function in the **starter code** and see if it works as expected.

Coding Task 2

The main function is being redesigned as follows:

```java
public static void main(String[] args){

	int[][] nums = getRadioData();
	displayRadioCalls(nums);

}

public static int[][] getRadioDat(){

	int[][] samples = new int[5][3]; //declare and initialise array size

	//code to populate samples not shown

	return samples;
}
```

The **`getRadioData`** function will populate a 5x3 2-dimensional array with values between 0 and 7 inclusive. It will **return** a reference to this array.

Implement the **`getRadioData`** function and rewrite the **main** function (see above) so that the program works as expected.

Coding Task 3

Design a new function **`getRowTotal`** that takes an integer representing a row as a parameter, and **returns** the total of numbers at that row index.

Its **signature** is:

```
Java
public int getRowTotal(int[][] radioCalls, int index){
	//to be implemented
}

//test with
public static void main(String[] args){
	int[][] nums = getRadioData();
	System.out.println(getRowTotal(3));


}
```

Coding Task 4

A function **`getMaxIndex`** processes a 2D array. It **returns** the **index** of the **row** with the highest total.

For example, in the sample 2D array above it will return 0 because row 0 has the highest total.

Its signature is:

```
Java
public static int getMaxIndex(int[][] radioCalls){
	//to be implemented
}
```

Call the function **`getRowTotal`** (Task 3) in your solution