Arrays and loops

Arrays
For loops
While
Do...While
Multidimensional arrays

Arrays

What if we have **more than one** variable of the same type, which could logically be grouped together?

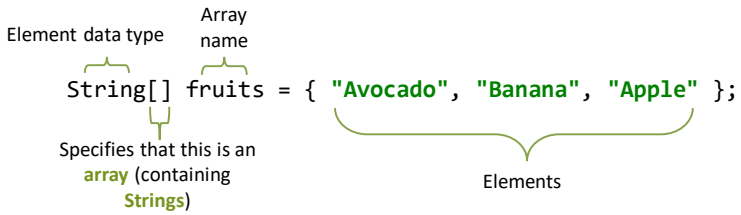
How you **don't** want to do it:

```
// One variable per row
String fruit1 = "Avocado";
String fruit2 = "Banana";
String fruit3 = "Apple";
…
```

Why don't we want to do it like this?

Anatomy of an array declaration

An array is a **data structure** that contains a number of variables called **elements.**



We have an array called **fruits** which contains variables of the type **String**. It contains three **elements**.

Anatomy of an array declaration

If we declare an array such as:

```
String[] fruits = { "Avocado", "Banana", "Apple" };
```

We get an array like this:

Index	Element
0	Avocado
1	Banana
2	Apple

fruits.length == 3

Arrays always starts counting from zero

Accessing elements

How do we access the elements?

If we declare an array such as:

```
String[] fruits = { "Avocado", "Banana", "Apple" };
```

We can access the elements through:

```
// Will output "Avocado" to the console
System.out.println(fruits[0]);
```

Zero-based index (0-2 in this case)

And modify an element through:

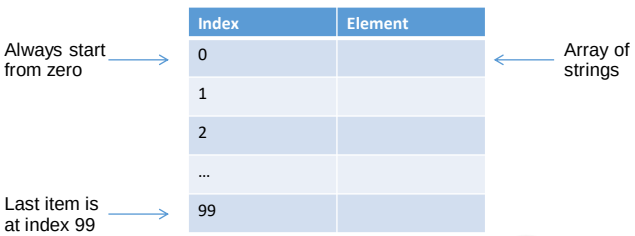
```
// Will replace "Apple" with "Coconut"
fruits[2] = "Coconut";
```

New value

Initializing an array

To create an empty array, we can write:

```
String[] fruits = new String[100];
```



Looping through arrays

Looping through arrays

How could we output every fruit in the array?

How you **don't** want to do it:

```
// One row per item
System.out.println("Fruit type: " + fruits[0]);
System.out.println("Fruit type: " + fruits[1]);
System.out.println("Fruit type: " + fruits[2]);
```

Why don't we want to do it like this?

for loops

We don’t want to modify our code every time we add elements to the array.

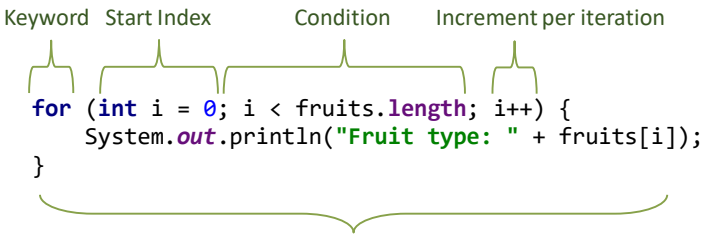
To get around this, we could use a **for loop**.

```
for (int i = 0; i < fruits.length; i++) {
    System.out.println("Fruit type: " + fruits[i]);
}
```

Notice that we can reach information such as **length** of the array by applying the **dot operator**.

Anatomy of the for loop (1)

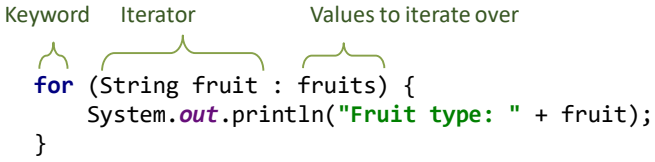
The **for** loop syntax.



Do this for every element, where i varies from 0 to fruits.length, incrementing by 1 with each iteration.

For each

Do we have any alternatives?




```
for (String fruit : fruits) {
    System.out.println("Fruit type: " + fruit);
}
```

This does exactly the same thing as the other for loop did, but we're just telling the compiler to do it with every object in the array, not between two certain indices.

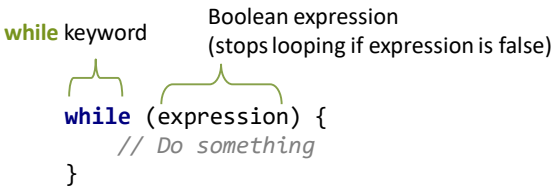
Tip: Get to know this syntax well. More often than not, this is the one you end up using over the other loop syntaxes

While loops

The while loop

Loop while a certain expression is true.

”While this statement is true, I want to...”



Using the while loop

Our example with the **for** loop could be rewritten using a **while** loop instead:

```
public static void main(String[] args) {
    String[] fruits = { "Avocado", "Banana", "Apple" };

    int index = 0;

    while (index < fruits.length) {
        System.out.println("Fruit: " + fruits[index]);
        index++;
    }
}
```

do ... while

The **do..while** loop is a variant of the **while** loop.

”I want to do [...] and then check if my statement still holds, and repeat if it does”

do keyword

```
do {
    // Do something while expression is true
} while (expression);
```

Boolean expression
(stops looping if expression is false)

do vs while: Differences?

The **while** loop checks the condition *first*, while the **do** loop checks the condition *afterwards*.

```
while (expression) {
    // Do something
}
```

```
do {
    // Do something
} while (expression);
```

What are the implications of this? How do the while and do ... while loops differ?

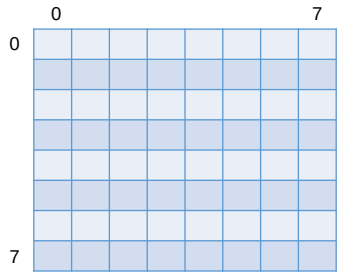
Multidimensional arrays

Multidimensional arrays

Arrays can have more than one dimension

```
//Create a 8x8 2-dimensional array
int[][] chessboard = new int[8][8];

System.out.println(chessboard[2][3]);
```



Multidimensional arrays

A 2D-array can be initialized and filled with data at the same time:

```
String[][] myStringArray = new String [][] { { "John", "Andersson"},
                                              { "Sven", "Johansson"},
                                              { "Mad", "Max"},
                                              { "Marie", "Nilsson"},
                                              { "Tore", "Svensson"} };
```

	0	1
0	John	Andersson
	Sven	Johansson
2	Mad	Max
	Marie	Nilsson
4	Tore	Svensson

```
System.out.println(myStringArray[2][0]);
System.out.println(myStringArray[2][1]);
```

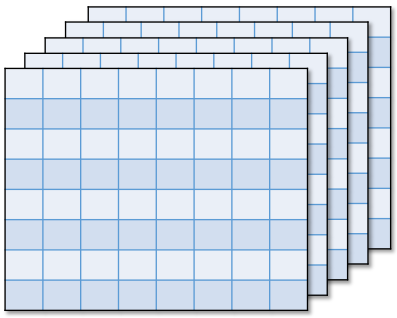
```
Mad
Max
```


Multidimensional arrays

We can have many dimensions, for example an array that can hold 4 chess boards:

```
//Create a 8x8x4 3-dimensional array
int[][][] chessgames = new int[8][8][4];

System.out.println(chessgames[2][3][1]);
```



Exercise 7

Let's do exercise 7