



# Welcome back to the Java Course

Module 2



# Array

An array is an ordered sequence of values of the same type.

```
int numbers[ ] = new int[5];
```

```
String names[ ] = new String[3];
```



# Array

```
type array_name[ ] = new type [ number ]
```



# Array

```
type array_name[ ] = new type [ number ]
```



# Array

```
type array_name[ ] = new type [ number ]
```



# Array

```
type array_name[ ] = new type [ number ]
```



# Array

```
type array_name[ ] = new type [ number ]
```



# Array

```
type array_name[ ] = new type [ number ]
```





## Array

array\_name [0] = val\_0

array\_name [1] = val\_1

...

array\_name [n] = val\_n



# Array

You can declare, create, and initialize an array in one line:

```
type array_name[ ] = {val_0 , val_1 , ... , val_n}
```



# List

In java list is basically an array with a dynamic size. It is a little bit more advanced tool.

```
ArrayList<String> stringList = new ArrayList<String>();
```



## Array vs List

### **Dynamic Size:**

Lists, such as ArrayList, can dynamically grow and shrink in size as elements are added or removed. They don't have a fixed size.



## Array vs List

### **Object Types Only:**

Lists can only store objects, not primitive data types. We'll see them properly with the Object Oriented Programming.



# Array vs List

## **Direct Access:**

You can directly access elements in an array using their index. This makes reading and writing elements very fast.



Let's practice!



# Searching

Searching refers to the process of finding a specific element (or determining its absence) into a collection of data.





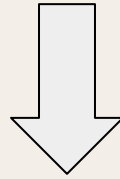
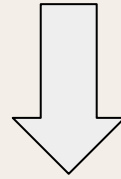
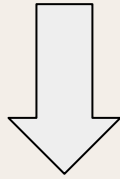
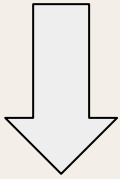
# Searching

How can we find a specific element in an array?



## Searching

array[0]   array[1]   array[2]   array[3]



value\_0   value\_1   value\_2   value\_3



# Searching

array [0] = "H"

array [1] = "e"

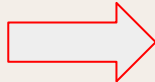
array [2] = "l"

array [3] = "l"

array [4] = "o"



# Searching

```
for (int i= 0; i< array.length ; i++){  
    array[i]        ???  
}
```



# Searching

```
for (int i= 0; i< array.length ; i++){  
    array[i]    ➡ "H" when i is 0  
}
```



# Searching

```
for (int i= 0; i< array.length ; i++){  
    array[i]    ➡  "e" when i is 1  
}
```



# Searching

```
for (int i= 0; i< array.length ; i++){  
    array[i]    ➡ "I" when i is 2  
}
```



## Searching

```
for (int i= 0; i< array.length ; i++){  
    if (array[i] == "e"){  
        // we found the value "e"  
    }  
}
```





Let's practice!



# Sorting

Sorting involves arranging elements in a specific order, such as ascending or descending.

You can sort both numbers and strings.



# Sorting

In java you can sort automatically an array using the method *sort()*. But remember to import the correct library.



# Sorting

```
import java.util.Arrays;
```

```
...
```

```
    Type array_name[] = {values};
```

```
    Arrays.sort(array_name);
```



## Sorting

```
import java.util.Arrays;
```

```
...
```

```
    Type array_name[] = {values};
```

```
    Arrays.sort(array_name);
```



## Sorting

```
import java.util.Arrays;
```

```
...
```


```
    Type array_name[] = {values};
```

```
    Arrays.sort(array_name);
```



## Sorting

```
int numbers[] = {2 , 3 , 1};  
Arrays.sort(numbers);
```

numbers            { 1 , 2 , 3 }



# Sorting

Reversed sorting:

```
Arrays.sort(array_name, Collections.reverseOrder());
```





Let's practice!