Arrays

ARRAYS

A collection of variables of the same data type

- An array in java is an object
- An array variable references a group of data
- The size of an array is fixed

NULL

The value of an object that references nothing

```
String text; // null
text = "some text"; // Address of "some text"
```

CREATING ARRAYS

```
dataType[] arrayName; // null

arrayName = new dataType[10]; // Address of the 10
elements of type datatype

dataType[] arrayName = new dataType[10];

int[] numbers = new int[20];
```

DEFAULT VALUES

When an array is created, its elements are assigned the following default values:

- `0` for the numeric primitive data types
- `\u0000` for char types
- `false` for boolean types
- `null` for reference types

ACCESSING ARRAY ELEMENTS

Use brackets and indices

```
int[] numbers = new int[5]; // {0, 0, 0, 0, 0}

numbers[0] = 5; // {5, 0, 0, 0, 0}

numbers[2] = 8; // {5, 0, 8, 0, 0}

numbers[4] = 10; // {5, 0, 8, 0, 10}
```

ARRAY INITIALIZERS

```
double[] numbers = {1.9, 2.9, 3.4, 3.5};

double[] numbers = new double[4];
numbers[0] = 1.9;
numbers[1] = 2.9;
numbers[2] = 3.4;
numbers[3] = 3.5;

double[] numbers;
numbers = {1.9, 2.9, 3.4, 3.5}; // ERROR
```

PRINTING ARRAYS

```java
int[] numbers = {5, 0, 8, 0, 10};

System.out.println(numbers); // ADDRESS

for (int i = 0; i < numbers.length; i++)
    System.out.print(numbers[i] + " ");
```

```
[I@7b23ec81
5 0 8 0 10
```

EXCEEDING ARRAY BOUNDS

The indices must be between 0 and length - 1

```
char[] chars = {'a', 'b', 'c', 'd'};

// Index -1 out of bounds for length 4
System.out.println(chars[-1]);

// Index 4 out of bounds for length 4
System.out.println(chars[4]);

System.out.println(chars); // abcd
```

PASSING ARRAYS TO METHODS

Arrays are passed by reference

```
public static void main(String[] args) {
    int[] numbers = {0, 1};
    change(numbers);
    printArray(numbers); // 1 0
}

public static void change(int[] numbers) {
    numbers[0] = 1; // {1, 1}
    numbers[1] = 0; // {1, 0}
}

public static void printArray(int[] numbers) {
    for(int i = 0; i < numbers.length; i++)
        System.out.print(numbers[i] + " ");
}
```

RETURNING ARRAYS FROM METHODS

```
public static int[] getNumbers() {
    int[] numbers = {1, 2, 3, 4, 5};

    return numbers;
}
```

ARRAYS CLASS

A class that contains some static methods that are used with arrays

- Sorting
- Searching
- Comparing
- Filling
- Returning a string representation of an array

SORTING ARRAYS

Using sort()

```
// sort(array): sorts the whole array
int[] numbers = {5, 2, 3, -1, 0, 4, 1};
Arrays.sort(numbers); // -1, 0, 1, 2, 3, 4, 5

char[] characters = {'a', 'z', 'b', 'w', 'c', 'A', 'D', 'Z', 'C'};
Arrays.sort(characters); // A, C, D, Z, a, b, c, w, z

int[] unicodes = {'a', 'z', 'b', 'w', 'c', 'A', 'D', 'Z', 'C'};
Arrays.sort(unicodes); // 65, 67, 68, 90, 97, 98, 99, 119, 122

// sort(array, fromIndex, toIndex): sort from (fromIndex) to (toIndex - 1)
int[] numbers = {5, 4, 3, 2, 1, 0, -1}; // 3, 6 + 1
Arrays.sort(numbers, 3, 7); // 5, 4, 3, -1, 0, 1, 2
```

SORTING ARRAYS

```
String[] strings = {"hij", "abc", "efg"};
Arrays.sort(strings); // abc, efg, hij


Point[] points = {new Point(1, 2), new Point(3, 4), new Point(-1, -2)};
Arrays.sort(points); // ClassCastException: class java.awt.Point
                        cannot be cast to class java.lang.Comparable
```

```
public static void main(String[] args) {
    String str = "";
    str.compareTo
}
```

compareTo(String anotherStrin... int
compareToIgnoreCase(String st... int

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip

SEARCHING ARRAYS

Using `binarySearch()`

- The array should be sorted in increasing order
- `binarySearch(array, element)`
  → *`binarySearch`*`(numbers, 4)`

Return values:
- Index of element inside the array if exists
- `-(insertionIndex + 1)` if the element was not found

Example: `{1, 2, 3, 5, 6, 7}` → `{1, 2, 3, 4, 5, 6, 7}`

SEARCHING ARRAYS

```
int[] numbers = {5, 4, 3, 2, 1, 0, -1};
Arrays.sort(numbers); // -1, 0, 1, 2, 3, 4, 5

System.out.println( Arrays.binarySearch(numbers, 4) ); // 5
System.out.println( Arrays.binarySearch(numbers, 3) ); // 4
System.out.println( Arrays.binarySearch(numbers, -3) ); // -1
System.out.println( Arrays.binarySearch(numbers, 6) ); // -8

String[] strings = {"a", "b", "c"};
System.out.println(Arrays.binarySearch(strings, "a")); // 0
System.out.println(Arrays.binarySearch(strings, "c")); // 2
System.out.println(Arrays.binarySearch(strings, "A")); // -1
System.out.println(Arrays.binarySearch(strings, "d")); // -4
```

COMPARING ARRAYS

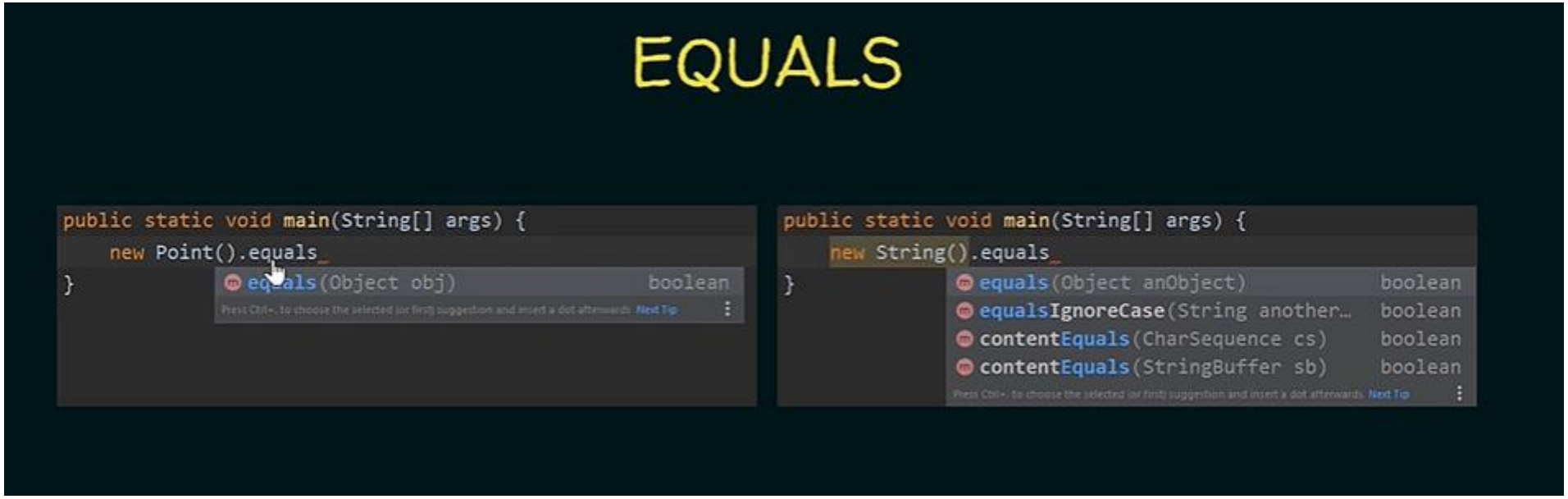
Using equals()

```
int[] numbers1 = {5, 4, 3, 2, 1, 0, -1};
int[] numbers2 = {5, 4, 3, 2, 1, 0, -1};
int[] numbers3 = {1, 2, 3, 7, 7, 8, 1};

System.out.println(numbers1 == numbers2); // false
System.out.println(Arrays.equals(numbers1, numbers2)); // true
System.out.println(Arrays.equals(numbers1, numbers3)); // false
```

COMPARING ARRAYS

```
String[] strings1 = {"a", "b", "c"};
String[] strings2 = {"a", "b", "c"};

System.out.println(strings1 == strings2); // false
System.out.println(Arrays.equals(strings1, strings2)); // true

Point[] points1 = {new Point(1, 2), new Point(3, 4)};
Point[] points2 = {new Point(1, 2), new Point(3, 4)};
Point[] points3 = {new Point(0, 0), new Point(3, 4)};

System.out.println(points1 == points2); // false
System.out.println(Arrays.equals(points1, points2)); // true
System.out.println(Arrays.equals(points1, points3)); // false
```

Equals method of the object



EQUALS

```
String str1 = new String("hello");
String str2 = new String("hello");

System.out.println( str1 == str2 ); // false
System.out.println( str1.equals(str2) ); // true

Point point1 = new Point(1, 2);
Point point2 = new Point(1, 2);

System.out.println( point1 == point2 ); // false
System.out.println( point1.equals(point2) ); // true
```

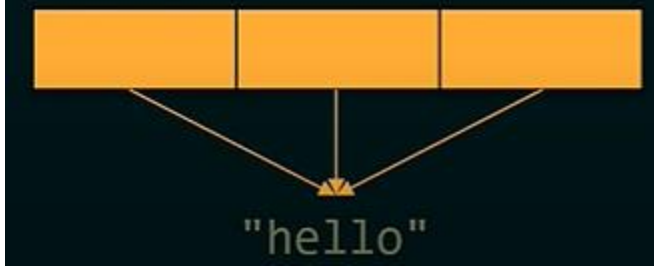
FILLING ARRAYS

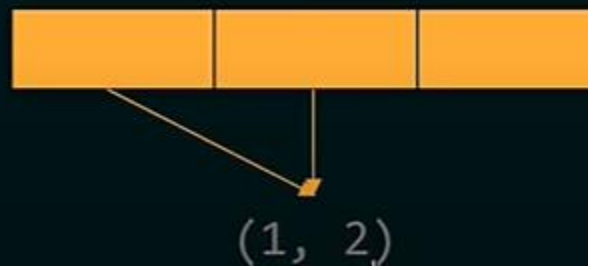
Using fill()

```
// fill(array, value): fill whole array
int[] numbers1 = new int[8]; // {0, 0, 0, 0, 0, 0, 0, 0}
Arrays.fill(numbers1, 3); // {3, 3, 3, 3, 3, 3, 3, 3}

// fill(array, fromIndex, toIndex, value)
int[] numbers2 = new int[8]; // {0, 0, 0, 0, 0, 0, 0, 0}
Arrays.fill(numbers2, 3, 7, 5); // {0, 0, 0, 5, 5, 5, 5, 0}
```

FILLING ARRAYS

```
String[] strings = new String[3]; // {null, null, null}
Arrays.fill(strings, "hello"); // {hello, hello, hello}

Point[] points = new Point[3]; // {null, null, null}
Arrays.fill(points, 0, 2, new Point(1, 2)); // {(1, 2), (1, 2), null}
```





PRINTING ARRAYS

Using toString()

```
System.out.println(Arrays.toString(strings));

System.out.println(Arrays.toString(points));

int[] numbers = {1, 2, 3, 4, 5};
System.out.println(Arrays.toString(numbers));
```

```
[hello, hello, hello]

[java.awt.Point[x=1,y=2], java.awt.Point[x=1,y=2], null]

[1, 2, 3, 4, 5]
```

Exercise 1

- Write a program that fills an array with n integers entered by an user
- Suppose the user can enter at least 1 number and at most 10 numbers

- Write a program that displays the sum, product and average of the elements of an integer array

- [1, 2, -3, 5, 7]
- Sum = 12
- Product= -210
- Avg = 2.4

Exercise 3

- Writing a program that displays the number of times/occurrences of an element in an array

- [1,1,1,2,3,4]
- The element 3 repeated only once
- The element 1 repeated 3 times
- The element 100 repeated 0 times