An [array in Java](https://www.geeksforgeeks.org/arrays-in-java/) is an object.

Now the question how is this possible?

What is the reason behind that?

Array is considered to be an object in Java. The reason behind this is that an array can be created using the ‘new’ keyword. The ‘new’ keyword/operator is always used to create an object. This is how an array is perceived as an object.

The direct parent class or super class of any array is the ‘Object’ class. Every array type in Java belongs to a certain class. This indicates that there are explicit classes for integer array types, float array types, double array types, and so on.

Arrays can be dynamically created, and be assigned variables as well.

Now the question also arises, every time we create an object for a class then what is the class of array?

* In Java, there is a class for every array type, so there’s a class for int [] and similarly for float, double etc.
* The direct superclass of an array type is Object. Every array type implements the interfaces Cloneable and java.io.Serializable.
* In the Java programming language, arrays are objects , are dynamically created, and may be assigned to variables of type Object . All methods of class Object may be invoked on an array.

**[What is the slight difference between object and array in Java?](https://www.quora.com/What-is-the-difference-between-object-and-array-in-Java" \t "_blank)**

Arrays are a native part of the Java language. Objects are types that are defined by the programmer and which inherit from Java.lang.object (or another one of its children.) Objects are defined by a Class definition, and like all instances of classes, can contain both *state*(data) and *behavior*(code, in the form of methods acting on the state and/or other data provided as parameters.). Arrays, in contrast, are defined in code using the special array syntax and keywords in the language, without having to define a class that creates the array’s capabilities.

Both share the property of being *reference* types, in that when passed as a parameter to a method, what is received is not a copy of the contents of the original object, but rather a reference to the array or object’sactual location in memory. This is why changes made to a reference type in a method are persistent even after the method ends, because it is the original item that is being altered and not a copy that gets disposed of once the method returns.