“Java is Strictly Pass by Value”

The following things to remember about object references: -

- Java always passes parameter variables by value.
- Object variables in Java always point to the real object in the memory **heap**.
- A mutable object’s value can be changed when it is passed to a method.
- An immutable object’s value cannot be changed, even if it is passed a new value.
- “Passing by value” refers to passing a copy of the value.
- “Passing by reference” refers to passing the real reference of the variable in memory.
- Primitive types are allocated in the stack memory, so only the local value will be changed. In this case, there is no object reference.

How about objects or references?

In Java, all primitives like int, char, etc. are similar to C/C++, but all non-primitives (or objects of any class) are always references. So, it gets tricky when we pass object references to methods.

Java creates a copy of references and pass it to method, but they still point to same memory reference.

Mean if set some other object to reference passed inside method, the object from calling method as well its reference will remain unaffected.

What is call by value and call by reference in Java?

Answer: - Call by value in java means passing a copy of the value to be passed.

Pass by reference in java means the passing the address itself.

In Java the arguments of the method are always passed by value.

With Java objects, the object reference itself is passed by value and so both the original reference and parameter copy both refer to the same Java object. Java primitives too are passed by value.