Jul 30 '11 at 7:23 | **Unknown author**

Is Java “pass-by-reference” or “pass-by-value”?

I just noticed you referenced my article.

The Java Spec says that everything in Java is pass-by-value. There is no such thing as "pass-by-reference" in Java.

The key to understanding this is that something like

```
Dog myDog;
```

is *not* a Dog; it's actually a *pointer* to a Dog.

What that means, is when you have

```
Dog myDog = new Dog("Rover");
foo(myDog);
```

you're essentially passing the *address* of the created `Dog` object to the `foo` method.

(I say essentially because Java pointers aren't direct addresses, but it's easiest to think of them that way)

Suppose the `Dog` object resides at memory address 42. This means we pass 42 to the method.

if the Method were defined as

```
public void foo(Dog someDog) {
    someDog.setName("Max");     // AAA
    someDog = new Dog("Fifi");  // BBB
    someDog.setName("Rowlf");   // CCC
}
```

let's look at what's happening.

- the parameter `someDog` is set to the value 42
- at line "AAA"
    - `someDog` is followed to the `Dog` it points to (the `Dog` object at address 42)
    - that `Dog` (the one at address 42) is asked to change his name to Max
- at line "BBB"
    - a new `Dog` is created. Let's say he's at address 74
    - we assign the parameter `someDog` to 74
- at line "CCC"
    - someDog is followed to the `Dog` it points to (the `Dog` object at address 74)
    - that `Dog` (the one at address 74) is asked to change his name to Rowlf
- then, we return

Now let's think about what happens outside the method:

Did `myDog` change?

There's the key.

Keeping in mind that `myDog` is a *pointer*, and not an actual `Dog`, the answer is NO. `myDog` still has the value 42; it's still pointing to the original `Dog` (but note that because of line "AAA", its name is now "Max" - still the same Dog; `myDog`'s value has not changed.)

It's perfectly valid to *follow* an address and change what's at the end of it; that does not change the variable, however.

Java works exactly like C. You can assign a pointer, pass the pointer to a method, follow the pointer in the method and change the data that was pointed to. However, you cannot change where that pointer points.

In C++, Ada, Pascal and other languages that support pass-by-reference, you can actually change the variable that was passed.

If Java had pass-by-reference semantics, the `foo` method we defined above would have changed where `myDog` was pointing when it assigned a `Dog` on line BBB.

Think of reference parameters as being aliases for the variable passed in. When that alias is assigned, so is the variable that was passed in.

community wiki

share|edit|flag edited Aug 28 '15 at 15:32 11 revs, 6 users 73% Scott Stanchfield

---