

0.1 Reference Example B

In the lectures we looked at an example that isn't in the notes. Some people have struggled to understand it and I hope this brief document will help explain it.

The code used was:

```
public class PassbyReferenceB {

    public static void swap (Person a, Person b) {
        Person ptemp = a;
        a=b;
        b=ptemp;
    }

    public static void main(String[] args) {

        Person p1 = new Person();
        p1.name = "Alice";
        Person p2 = new Person();
        p2.name="Bob";
        System.out.println("P1: "+p1.name+" "+"P2: "+p2.name);

        Person ptemp = p1;
        p1=p2;
        p2=ptemp;
        System.out.println("P1: "+p1.name+" "+"P2: "+p2.name);

        swap(p1,p2);
        System.out.println("P1: "+p1.name+" "+"P2: "+p2.name);

    }
}
```

There are three parts to this example: we'll look at each in turn.

0.1.1 Part I

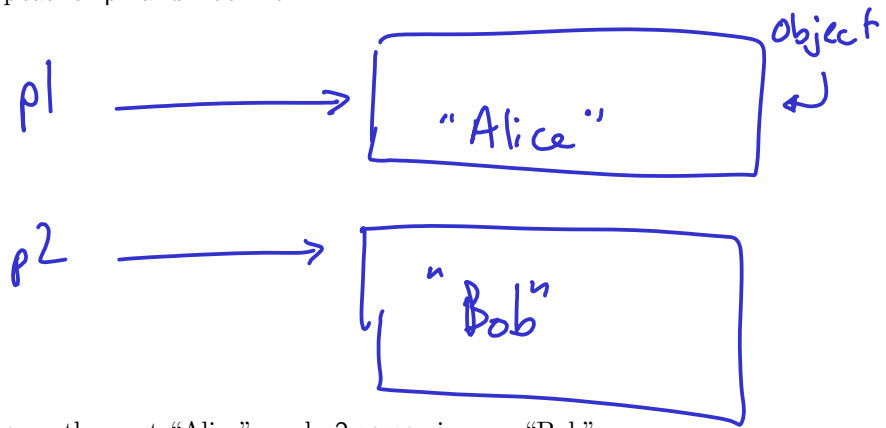
```
Person p1 = new Person();
p1.name = "Alice";
Person p2 = new Person();
p2.name="Bob";
System.out.println(p1.name + " " + p2.name);
```

The first line here does two things and we can emphasise it by splitting it into two further lines:

```
Person p1;  
p1 = new Person();
```

So, firstly it declares a *reference* called `p1`. In the split example, `p1` is *uninitialised*; that is to say that it doesn't point anywhere. In Java, this means that `p1` is `null` and we can test for that if we wanted to (if `(p1==null)` ...).

The second bit (`p1 = new Person()`) creates an object of type `Person` and sets the reference `p1` to point to it. Then we use the reference `p1` to get to the object and mutate its state, setting its name to Alice. Then we repeat for `p2` and Bob. i.e.

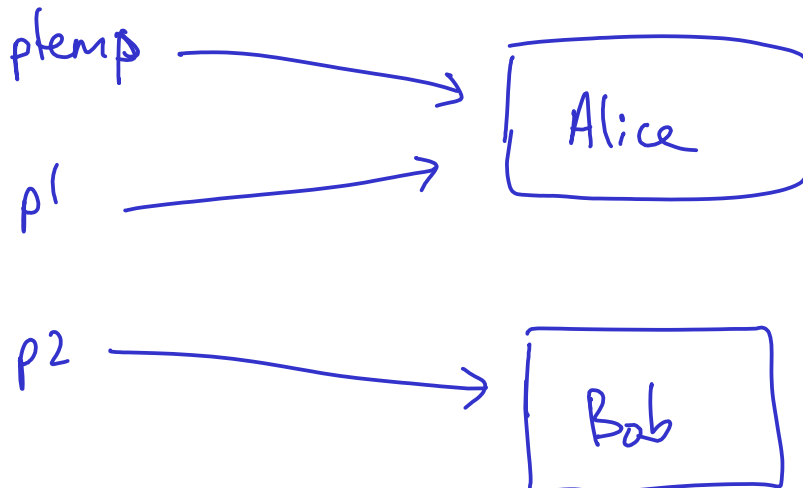


When we print out `p1.name` we then get "Alice", and `p2.name` gives us "Bob".

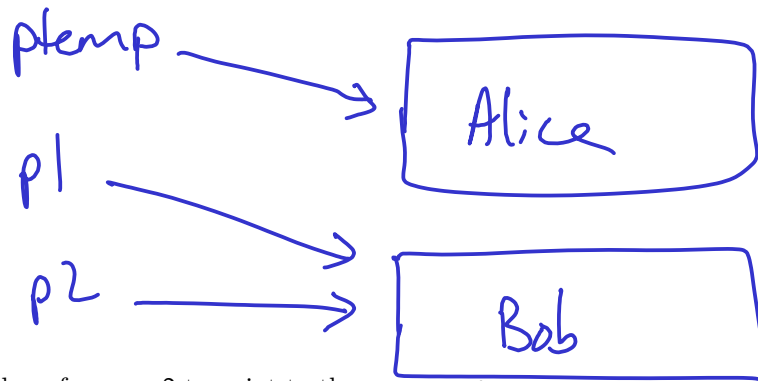
0.1.2 Part II

```
Person ptemp = p1;  
p1=p2;  
p2=ptemp;  
System.out.println(p1.name + " " + p2.name);
```

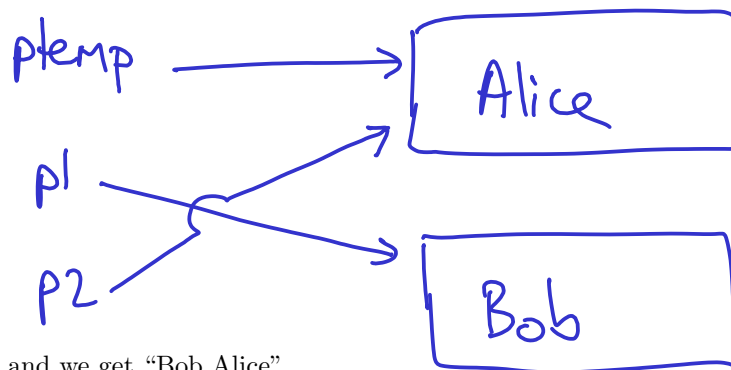
Here we create a new reference called `ptemp` and point it to the same thing as `p1`:



Now we set the reference p1 to the same as p2 (p1=p2):



Finally we set the reference p2 to point to the same as ptemp:



Now, we print and we get "Bob Alice".

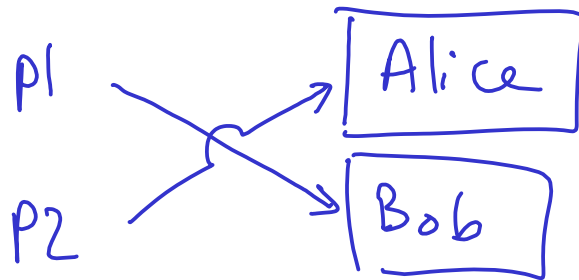
0.1.3 Part III

```
public static void swap (Person a, Person b) {  
    Person ptemp = a;  
    a=b;  
    b=ptemp;  
}
```

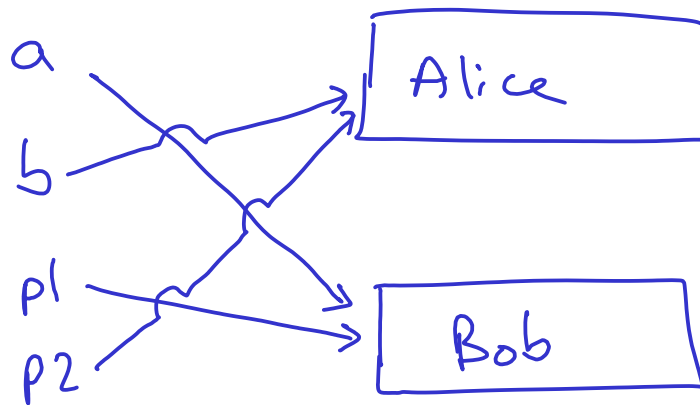
```
...  
swap(p1,p2);  
System.out.println(p1.name + " " + p2.name);  
...
```

At first glance, the code in the function swap() looks like the code we just looked at in Part II, so it's tempting to say that the references get swapped over. Let's look more closely. We start

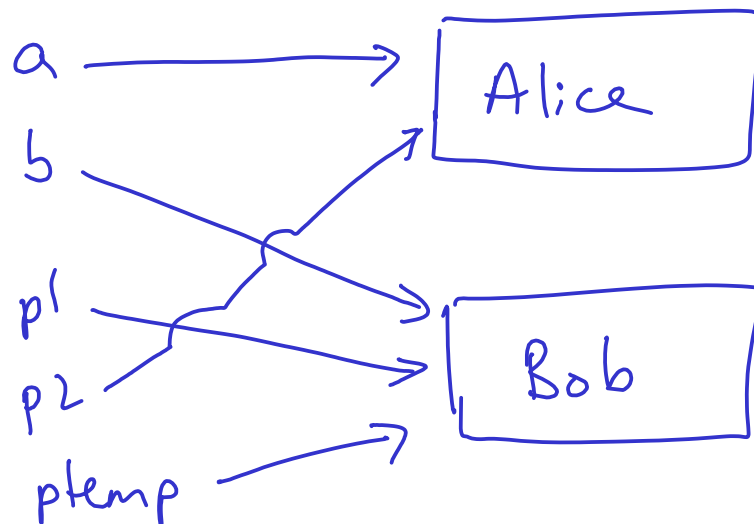
here:



we call the `swap()` function and, inside this function, there are two references to Person objects created, `a` and `b`. **Java copies the arguments (“call by value”)** and so **`a` and `b` end up as copies of `p1` and `p2` respectively.** Since we copy the *reference* and not the *object* we get:

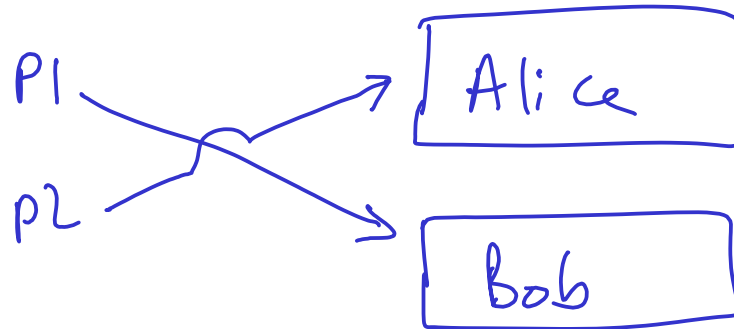


Then we create a reference `ptemp` and we swap over `a` and `b` just like in Part II, giving us at the end of the `swap()` function:



only `a, b`
swapped.
`p1, p2` as
before.

Now that function is finished and its 'local' variables (everything it introduced) are destroyed, leaving:



And so `swap()` was never able to touch the references `p1` and `p2` and it's left in the same state as it started Part III in. i.e. it prints "Bob Alice" again.