

# Re: Passing NULL Objects

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.help/2005-11/msg00168.html>

---

- *From:* fb <[fb@xxxxxxxxxx](mailto:fb@xxxxxxxxxx)>
  - *Date:* Fri, 04 Nov 2005 18:03:57 GMT
- 

Rhino wrote:

"fb" <[fb@xxxxxxxxxx](mailto:fb@xxxxxxxxxx)> wrote in message  
[news:wsAaf.402999\\$t12.182196@xxxxxxxxxxxxx](mailto:news:wsAaf.402999$t12.182196@xxxxxxxxxxxxx)

Hello everyone. I have the following:

```
//Perform a search for the account number
public BankAccount searchAccounts (Vector<BankAccount> bankAccounts ,
long accountNumber){
    int accNum=0;
    for(int x=0;x<bankAccounts.size();x++){
        if(bankAccounts.elementAt(x) == accountNumber){
            return ?? //I want to return the object that matches from the
Vector to the calling function
        }
        else
            return ?? //How do I return a NULL object (The reason is that I
can test for it in the returning function)
    }
}
```

So...returning null objects. Any thoughts? :-)

```
Usually, returning a null is easy:
---
return null;
---
```

Thanks!

## Re: Passing NULL Objects

Or, more typically, you declare the object that is being returned and initialize it to null at the beginning of the method. Then, the code in the rest of the method tries to assign a value or set of values to the object being returned but, if that fails, hands back the null object. For example:

```
---
public Integer searchAccounts(Vector bankAccounts) {

    Integer accountNumber = null; //initialize object to be returned to null

    //Determine int version of account number based on data in incoming Vector
    int acctno = ....

    return new Integer(acctno); [if no account number could be determined,
    return a null]
}
---
```

Your situation is a bit more difficult since you can't assign a null to an int value, which is a primitive, only to an object. So, you can either change your logic to return an object - Integer would be a logical choice, in which your code would look a lot like the example I just gave - or, you can do what I often do in methods: return an Object array. The first thing in the Object array is always a boolean which is set to true if the method returned what it was supposed to find or set to false if the method encounters a problem. Also, if the method found what it was supposed to find, return the thing you were trying to find along with the boolean true; if the method failed, return a boolean false and the error message that explains what went wrong. For example, if the code is looking up the customer's account number and gets a bad return code from the database lookup, I'd pass back a false in the boolean, followed by the error message describing the problem. That would look something like this:

```
---
public Object[] findCustomerName (int accountNumber) {

    String customerName = null;
```

## Re: Passing NULL Objects

```
//Look up customer info based on account number
try {
customerName = ....
} catch (SQLException excp) {
    Object[] Result = {new Boolean(false), excp.getMessage()};
    return Result;
}

//If we get this far, the customer name was found.
Object[] Result = {new Boolean(true), customerName};
return Result;
}
```

```
---
Then, to invoke the method and check the result:
---
String custname = null;
Object[] Result = findCustomerName(12345);
boolean nameFound = ((Boolean) Result[0]).booleanValue(); //find out if
method worked
if (nameFound) {
    custname = (String) Result[1];
} else {
    String msg = (String) Result[1];
    //do appropriate error handling - display error message? Write it to a
log?
}
---
```

That probably looks like extra work but it makes it possible for your method to tell the invoker what specifically went wrong if there is a problem. Although you COULD just pass back a null or some other value and possibly infer the same thing, this approach makes it possible for you to tell the invoker WHY something failed, which can improve your program's error handling.

Rhino

Excellent example, I'll consider it closely. Ty.

fb

--

```
int main(void){char
p[]="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz.\ \n",*q="kl
BIcNBFr.NKEzjwCIxNJC";int i=sizeof p/2;char *strchr();int putchar(\
```

## Re: Passing NULL Objects

```
);while(*q){i+=strchr(p,*q++)-p;if(i>=(int)sizeof p)i-=sizeof  
p-1;putchar(p[i]\ );}return 0;}
```

.