

Forum:[Hack Forums](#)

Thread: [public boolean equals\(Object obj\)](#)

Stinger910 - October 6, 2015, 1:43 pm

I've got double variables and I could of sworn that you do it this way.

```
public boolean equals(Object obj)
{
return length == obj.length && height == obj.height && width == obj.width;
}
```

error message

Error: cannot find symbol

symbol: variable length

location: variable obj of type java.lang.Object

Any and all help is appreciated! Stuck on this one method!

Raijin - October 6, 2015, 1:45 pm

I am confused. What are you trying to return?

It must be a Boolean value that's returned (true/false).

Edit: == is used in conditional statements

Edit2: ignore me I'm a noob

TheCyberBrick - October 6, 2015, 1:58 pm

(Oct 6, 2015 01:45 PM)Raijin Wrote: [\[CITE\]](#) I am confused. What are you trying to return?

It must be a Boolean value that's returned (true/false).

Edit: == is used in conditional statements

It does return a boolean value and the usage of '==' is correct.

OP, your problem is that you're not casting the object. Your 'equals' method takes 'Object' as an argument, hence the program does not know exactly what is being passed as an argument because in java any class implicitly extends 'Object' (and therefore any object is an instance of 'Object'). Since the program doesn't know what kind of object it is, it can't access the fields 'length', 'height' and 'width' because it could be any other object that doesn't have those fields.

To fix your 'equals' method you have to cast the object. First off, if the object is not an instance of your class you can already return false. If it is an instance of your class, then you cast the object and perform the check. This can be achieved with this code:

```
Code public boolean equals(Object obj) {
    if(obj instanceof ThisClass) {
        //Cast object so that the program knows what kind of object it is
        ThisClass castObject = (ThisClass)obj;
        //Now you can perform the check
        return length == castObject.length && height == castObject.height && width ==
castObject.width;
    } else {
        //Not an instance of ThisClass, can't be equal
        return false;
    }
}
```

}

-Base- - *October 6, 2015, 4:55 pm*

The only addition I'd make to TheCyberBricks post is that you shouldn't use the instanceof keyword when implementing the equals method because then classes that are subclass of the class will be true for that condition. Instead check the direct class of the objects and see if they're equal.

Natecat33 - *October 6, 2015, 4:58 pm*

(Oct 6, 2015 04:55 PM)-Base- Wrote: [\[CITE\]](#) The only addition I'd make to TheCyberBricks post is that you shouldn't use the instanceof keyword when implementing the equals method because then classes that are subclass of the class will be true for that condition. Instead check the direct class of the objects and see if they're equal.

You actually should use the instanceof in this situation. All subclasses will have those variables. The entire point of passing in an object instead of the actual class is so that you can include subclasses and not have to do a cast everytime you call that method.