

Forum:[Hack Forums](#)

Thread: [Newbie Java Question\(s\)](#).

?Saadi - June 19, 2018, 9:04 pm

Hello, I've just started learning Object-Oriented Programming in Java and am trying to get my basics right.

Anyways, had a couple of quick questions on "Classes & Objects":

--

Say a Class, which has only the main method:

Inside the main method, when we create new instances of the type of some other Class, are these "instances" known as "objects" or are they known as "variables"?

Moreover, when these 'instances' are 'made', where are they stored? The class with the main function, or the 'other' class with the constructor? What are the 'scopes' of these instances?

Thank you!

P.S; if this is the wrong forum, my apologies.

Xamarin - June 20, 2018, 2:38 am

Hello! Well ... I am not very good at explaining, but I will do my best to do so.

When we create new instances of a class inside a main method, or any method for that matter, then the result is a new instance of the requested object. The "variable" that holds this instance of the object, is actually a pointer to the instance of an object. For example, consider the following code:

```
Code public class SampleMainClass
{
    public static void main(String[] args)
    {
        Dog myDog = new Dog();
    }
}
```

Here we have created a new instance of an object of type Dog, with a name myDog. This new instance of the object, is saved in memory. If it helps you, consider the myDog "variable" as a pointer to the newly created object; it points to the memory region where this instance of the object is. In general though, we will refer to myDog as being an object, and not a variable.

Now, about the scope. The instance of the object in the above code, will last until the end of the method; then it will get destroyed by the garbage collector. In general, consider the code to be in levels. If you create a new instance of an object inside a method, this new instance will get destroyed when the method ends. If you instantiate an object in a class like this for example:

```
Code class SampleClass
{
```

```
AnotherClass anotherClassName = AnotherClass();  
}
```

then, the anotherClassName object will "live", until SampleClass stops existing, whenever that may be.

I hope I helped you a bit. Feel free to ask whatever else you want, and good luck on your programming-journey!

Tenno - June 20, 2018, 10:50 am

You can also create one instance of a class and use it in multiple classes by declaring the instance as static.

Example:

Code public class ClassA {

```
    public static void main(String[] args)  
    {  
        // Calling setNumber() from ClassB.  
        ClassB.inst.setNumber(3);  
  
        // Calling getNumber() from ClassB  
        System.out.println(ClassB.inst.getNumber());  
    }  
}
```

}

public class ClassB {

```
    private int number;
```

```
// Creating a new instance within the class itself. Declared as static.  
    public static ClassB inst = new ClassB();
```

```
    public void setNumber(int number){  
        this.number = number;  
    }
```

```
    public int getNumber(){  
        return this.number;  
    }
```

}Output: 3

Notice how we have not created an instance of class B in our main function, but we can still call class B methods. When using the static keyword, only one instance of the object is created and stored into memory. A static variable (in this case the object pointer) isn't pointing to a specific instance of a class, but the class itself. So when the class is loaded at runtime, the instance of the static variable is created and stored in memory. Since it's more on a class level than on an instance level, other classes can use it without any pointers to the class.

I'm not the best at explaining and not a very experienced Java programmer, so doing your own

research is recommended. Corrections also welcome :)