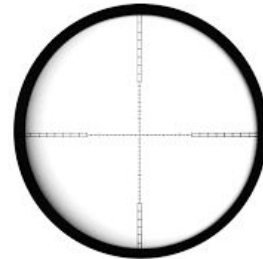
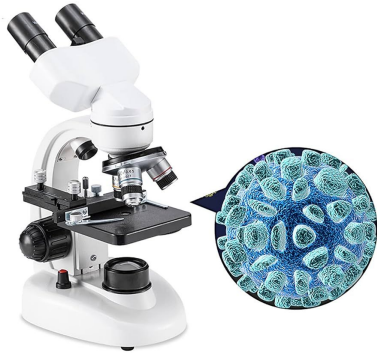


Scope

What is scope?

Scope is when a variable is only accessible inside the region it is created.

In other words, when you can and can't access a variable!



Easiest example of Scope

```
System.out.println(x);
```

```
int x = 5;
```

```
System.out.println(x);
```

X can't be accessed here!

X can be accessed here!
After it's created!

The Garbage Collection

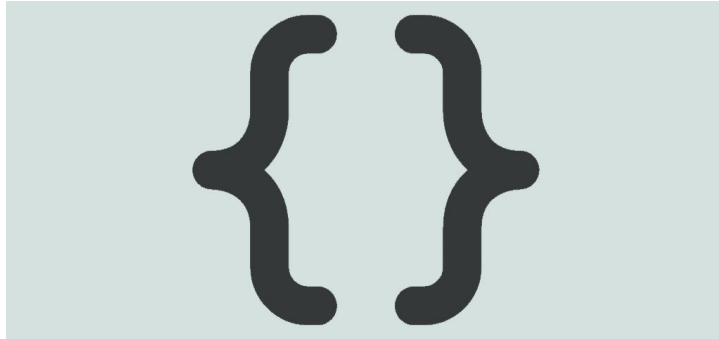
Java has a garbage collection system that automatically works and makes our lives MUCH easier!

The garbage collection does something called “sweeping” which is making more space in memory for other objects.



When does sweeping happen?

Curly brackets encompass code. The garbage collector sweeps at the end of every set of curly brackets. It deletes any and ALL variables created within the scope of the curly brackets.



Examples of Scope

```
if(4 > 3){  
    int answer = 4*3;  
}  
System.out.println(answer);
```

What is the scope of answer here?

Is this an error?

Examples of Scope

```
if(4 > 3){  
    int answer = 4*3;  
}  
System.out.println(answer);
```

Here answer is created
WITHIN the if statement.

The **scope** of **answer** is only
within the if statement

This code would provide the
error below.

```
starter.java:13: error: cannot find symbol  
        System.out.println(answer);  
                           ^  
symbol:   variable answer  
location: class starter  
1 error
```

Solutions to scope issues

1. Re-use old variables!
2. Construct variables outside of curly bracket zones
 - a. Variables must be constructed & declared WITH a default value
 - b. Then re-declare the constructed variable within the curly brackets

Solutions to scope issues

```
int answer;  
if(4 > 3){  
    answer = 4*3;  
}  
System.out.println(answer);
```

Here we construct “answer” above the brackets so its scope is within the entire file.

Here we just re-declare “answer” as $4*3$.

answer stays the value 12 even after the if statement

Compiling branches with Java

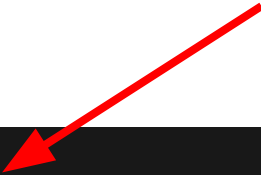
The compiler doesn't look at any data.

This means that ALL branches are compiled with the worse case scenario, FALSE on every case.

```
int answer;  
if(false){  
    answer = 4*3;  
}  
System.out.println(answer);
```

So if, the if statement
doesn't run, what is the
value of answer?

Solution: Give variables default values just in case



```
int answer = 0;
if(4 > 3){
    answer = 4*3;
}
System.out.println(answer);
```

Just in case the if statement doesn't run, the value of "answer" will be 0.

Lab: Scope

Fix errors in the given file to match the labOutput.