**What is scope?**
Scope indicates where in your program a variable can be used. Some variables can only be used inside a particular method, while others can be used anywhere in your class. When you are out of a variable's scope, it is not usable.

**Why is scope important?**
Scope impacts all of your variables, and you have some control over it. Scope constrains access to variables so that you know that only certain regions of your program can read their values or make changes to them. If you attempt to use a variable outside of its scope, the compiler will throw an error and your program will not be compiled into an executable.

**What kinds of scope are there?**
There are three major kinds, method, instance, and class. Below is a description of each and an example of what they look like.

**Method Variables:** Variables that exist only while a method is being run. Method variables can only be accessed inside the method in which they were created. Parameters are also method variables, the only difference is that their value is set when the method is invoked. Of all the types of variables, method variables have the shortest lifespan. They are accessed by just their name.

**Instance Variables:** Variables that exist as long as class instance does, instance variables can be used in any method and are accessed by just their name if you are in the same class. Instance variables can be used by child classes provided the variable was declared with the keyword `protected` next to it. Instance variables are put after the class variables at the top of your class file.

**Class Variables:** Variables that exist as long as program is running, denoted with the keyword `static`. They can be used in any method and are accessed by using the class name, followed by a dot, and then the name of the variable. Class variables are typically found at the top of your file just after the opening brace for the class (if any exist).

```java
public class MyClass {
    private float instanceVar = 3.0;
    private static int classVar = 4;
    ...
    public void setValue(int methodParam) {
        float methodVar = 0.0;
        instanceVar = methodVar + methodParam; // all are "in scope" here
        classVar++; // don't need to say MyClass.classVar since already in class
    }
    public float getValue() {
        float a = methodVar + methodParam; // bad: both "out of scope", only exist in setValue
        return instanceVar; // instanceVar is "in scope"
    }
}

public class OtherClass {
    ...
    public float doSomething() {
        float a;
        instanceVar = methodVar + methodParam; // bad: all three are out of scope
        a = classVar; // bad: classVar is not a part of OtherClass
        return MyClass.classVar; // classVar is a static (always existing) part of MyClass
    }
}
```
What is scope?
Scope indicates where in your program a variable can be used. Some variables can only be used inside a particular function, while others can be used anywhere in your class. When you are out of a variable's scope, it is not usable.

Why is scope important?
Scope impacts all of your variables, and you have some control over it. Scope constrains access to variables so that you know that only certain regions of your program can read their values or make changes to them. If you attempt to use a variable outside of its scope, the compiler will throw an error and your program will not be compiled into an executable.

What kinds of scope are there?
There are two major kinds, local and global. Below is a description of each and an example of what they look like.

**Local Variables:** Variables that exist only while a function is being run. Local variables can only be accessed inside the function in which they were created. Parameters are also function variables, the only difference is that their value is set when the function is invoked. Of all the types of variables, local variables have the shortest lifespan. They are referred to by just their name.

**Global Variables:** Variables that exist as long as program is running. They can be accessed anywhere in the program after they have been declared. Global variables are typically found at the top of your file before the main. Like local variables, they are referred to by their name.

```c
#include <math.h>

double globalVar = 0;
#define PI 3.14159
enum {RED = 0, BLUE, GREEN, ORANGE = 5, PURPLE};

int addValue(int anInt);

int main()
{
    int localVar = 7;
    globalVar = BLUE + localVar + 5.5 * PI; /* good: all are "in scope" here */
    globalVar = globalVar + anInt; /* bad: anInt is local to addValue function */
    localVar = addValue(localVar);
    return 0;
}

int addValue(int anInt)
{
    globalVar = globalVar + anInt; /* good: both are "in scope" */
    return localVar; /* bad: localVar is local to main */
}
```