Scope of variables. Methods in Java.

♦ In Java variables cannot be accessed outside of the code block { } in which they are created *(unless they have the public modifier in front of them)*

♦ A variable is destroyed when the program reaches the closing } of the code block in which it was declared.

```
public static void main(String[] args) {
	int x = 9;
	System.out.println(x);
} //x is destroyed here
```

Here's another example:

```
public static void main(String[] args) {

	if (1 < 2) {
		int x = 9;
		System.out.println(x);
	}

	x++; //ERROR: x cannot be found. It doesn't exist

	for (int i = 0; i < 10; i++) {
		System.out.print(i);
	}	//i is destroyed here

	System.out.println(i);  //ERROR: i does not exist.

	int j;

	for (j = 0; j < 5; j++) {
		System.out.print("*");
	}

	System.out.println(j);  //there is no problem accessing j here
}
```

This also works in methods:

```
public static void main(String[] args) {
	String food = "cake";

	method1();

	method2(food);	//we can pass the variable to the method

} //varible 'food' is destroyed here

static void method1() {
	System.out.println("Let them eat " + food);
	//ERROR: variable 'food' cannot be found
}

static void method2(String str) {
	System.out.println("Let them eat " + str);
	//prints "Let them eat cake"
}
```

There are other notes explaining how methods work.
http://quarkphysics.ca/ICS3U1/unit3/method2.htm
http://quarkphysics.ca/ICS3U1/unit3/method3.htm

Unfortunately there are a whole lot of things to learn when you first learn to program.
Java has a huge learning curve at the start.

Sometimes it's best to just start programming and then figure out what everything means and how it works later on.

Local and global variables

♦ Global variables can be accessed by all methods in that class.
♦ Global variables are automatically initialized to a default value (e.g. 0 or "")
♦ Local variables must be initialized before you can use them.

```
public class Temp2 {

	//GLOBAL VARIABLES
	static int y;		//y is set to zero

	public static void main(String[] args) {

		y = y + 10;		//no problem here as y is automatically set to zero

		int z;		//z has no value!

		z = z + 10;	//ERROR: local variable z has not been initialized
	}
}
```

Try this program:

```
public class Temp2 {

	//**** GLOBAL VARIABLES *****
	//static
	static int num = 5;
	//instance
	int length = 24;

	public static void main(String[] args) {
		System.out.println("global num = " + num);

		num *=100;
		System.out.println("global num = " + num);

		System.out.println(length);
		//ERROR: Cannot make a static reference to the non-static field 'length'

		//this is a local variable since it's declared inside a method.
		int num = -66;

		System.out.println("local num = " + num);
		num--;
		System.out.println("local num = " + num);

		System.out.println("global num = " + Temp2.num);
	}
}
```

Summarizing the previous examples:

♦ A static method cannot access anything that is not static. (Unless you create an object, using "new ___")
♦ Local variables will **shadow** global variables that have the same name.
♦ Global variables can always be accessed by typing the classname first: e.g. Temp2.num, Math.PI .