Here’s an example of how a class is initialized.

public class Person {

{

personCount = 10;

}

static {

personCount = 20;

}

private static int personCount = 1000;

{

personCount = 30;

}

static {

personCount = 40;

}

public static void main(String[] args) {

System.out.println(Person.personCount); // Line A

Person p = new Person(); // Line B

System.out.println(Person.personCount); // Line C

Person.personCount = 100; // Line D

p = new Person(); // Line E

System.out.println(Person.personCount); // Line F

}

}

When the program starts and this main method is kicked off, the Person class is loaded.  This, in turn, makes the static personCount variable available, with a default value of 0 (remember, all instance and static variables start with default values).

Next all of the static initialization blocks, and the static variable's explicit initialization are executed in the order they are listed, from top to bottom.  Therefore, the personCount variable is set to 20, 1000, and finally 40.  So Line A prints "40."

On Line B, an object is instantiated, so that causes the instance initialization blocks to be executed in the order they are listed, from top to bottom.  So now personCount is changed to 10 and finally 30.  Line C prints the value "30."

Line D changes the personCount to 100, but Line E causes the instance initialilzation blocks to be executed again.  Therefore, the final value printed in Line F is, once again, "30."  So the total output is:

40

30

30

**Next Lecture**

In the next lecture, we are taking a closer look at the stack (in memory).  In particular, we will talk about how a frame is added to the stack to hold local variables.  During this discussion, to help clarify the concept, I am writing out what is in the stack above the method in a comment, i.e.:

*// Customize frame: newColor = "blue"     newType = "convertible"*

I want to clarify that the comment is NOT code that is creating the actual frame.  The frame is automatically added to a stack when you invoke a method.

It's just a comment so you easily see the variables and values we are discussing, without causing a compile error.  As a reminder, a comment is "human readable text" intended for the developers looking at the code... it is not executable code for the program, and therefore won't throw any compile errors.