
This video explains the spacing and alignment of source code.

## What is indenting?

```
if (weather.equals("sunny"))
{
    out.println("Sunny days are fun for walking in the park.");
}
else if (weather.equals("cloudy"))
{
    out.println("Cloudy days are great for portrait photos.");
}
else
{
    out.println("You didn't enter cloudy or sunny.");
    out.println("I can't see from inside your computer.");
}
```





So what is indenting? Indent means move to the right. Let's take a look at a code example.

This if statement has 3 blocks of code, each inside an open and close curly brace pair. Inside each block, the print statements are indented - that is, they're moved towards the right. They have some spaces before them. <click>

We move these lines to the right so that it's easy to see the curly braces that start and end that block of code. It is also a clue to our eyes that the code inside the block only runs sometimes, not all the time. That is, it only runs when the if statement says it should. The spacing is a visual hint to make us notice that.

Notice how the word if, the curly braces, and the word else are all lined up in a straight line. <click>

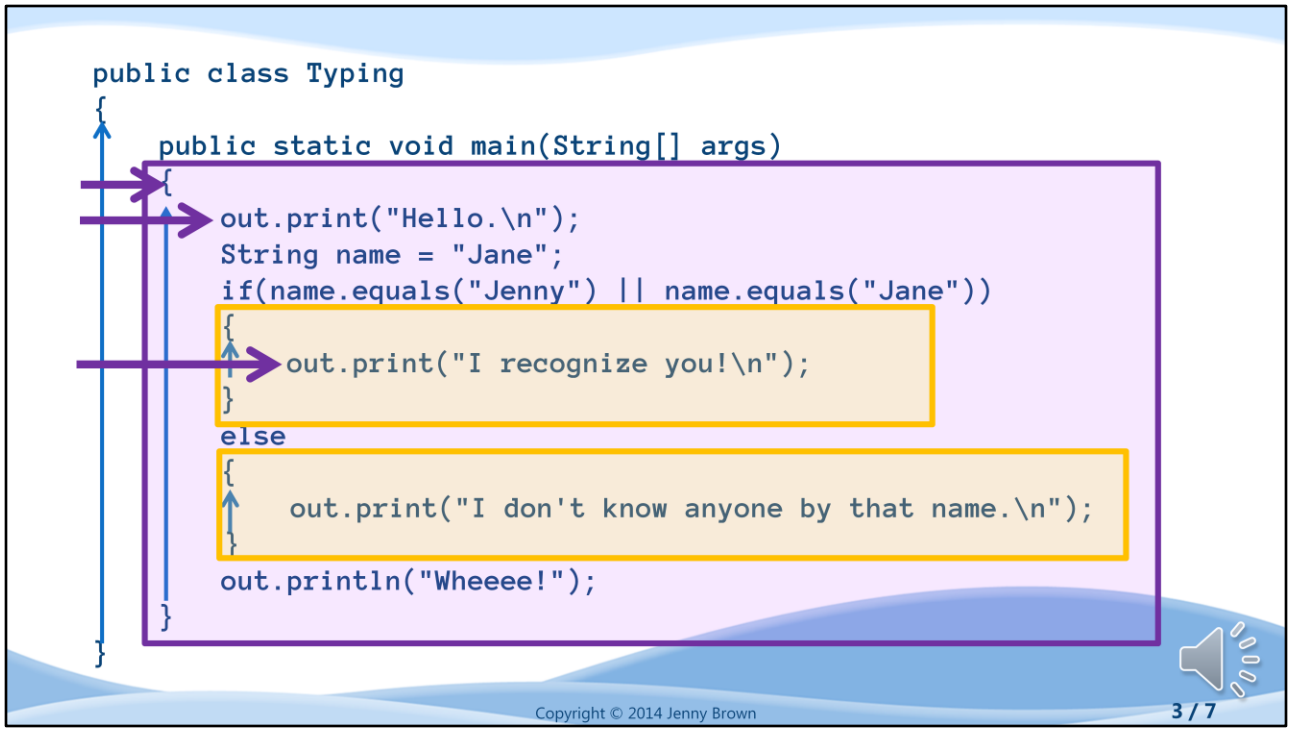
Eventually, we will be putting if statements inside other if statements, and the extra curly braces can look a little crazy. Keeping them carefully lined up like this makes it easier to find the matched pairs. I'll give an example of that in another slide.

Also notice how the out.println statements are all nicely lined up. <click> This makes it easier to see that they are at the same level of nesting, that is, that they are just inside one if statement. It makes the code easier to read.

Indenting is usually done by pressing the Tab key on the keyboard. You can use the

spacebar but it's a little more tedious to get it lined up just right.  The tabs are more consistent.

Let's take a look at a more complex example.

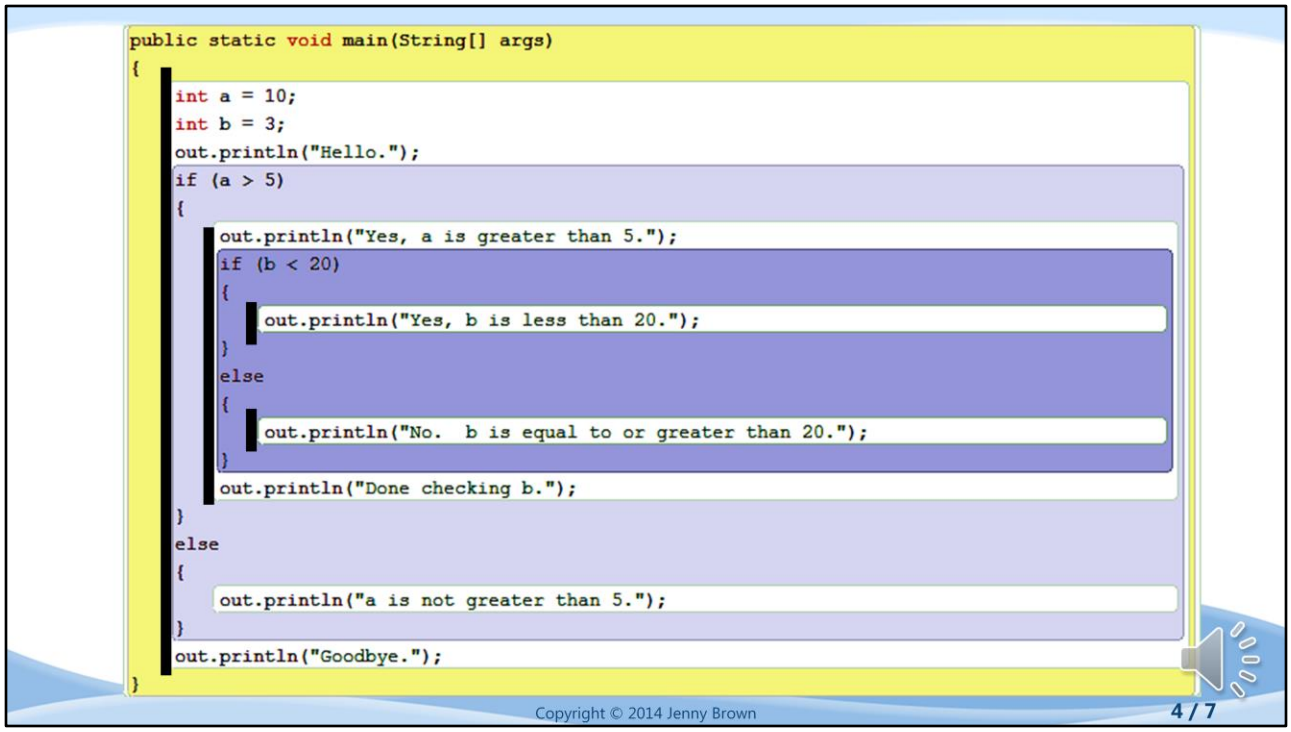



In this example, notice how the pairs of curly braces make perfect vertical lines. When you look at a close curly brace, it's easier to find the paired open curly brace. <click, click, click, pause, click>

Being able to look from the closed curly brace to the open curly brace tells you important things. It tells you what portion of the code belongs to that pair.

For instance, here's a set of curly braces and a highlight of the code that goes with it. <click> It belongs to the beginning part of the if statement.
Here's another set of curly braces and the matching code. <click> It belongs to the "else" part of the if statement.
Those are pretty short code blocks. In contrast, <click> here's the block of code that belongs to the main() method. Look at how the curly braces and indenting make it visually apparent what part is inside that block.

Also notice that there are three levels of indenting here. <click> Each time a set of curly braces gets started, the stuff inside it moves one step to the right. When the block of code is done and we close the curly brace, the next statement comes at the same level as the curly brace just closed. Look at the line that says "Wheee" for an example of that.



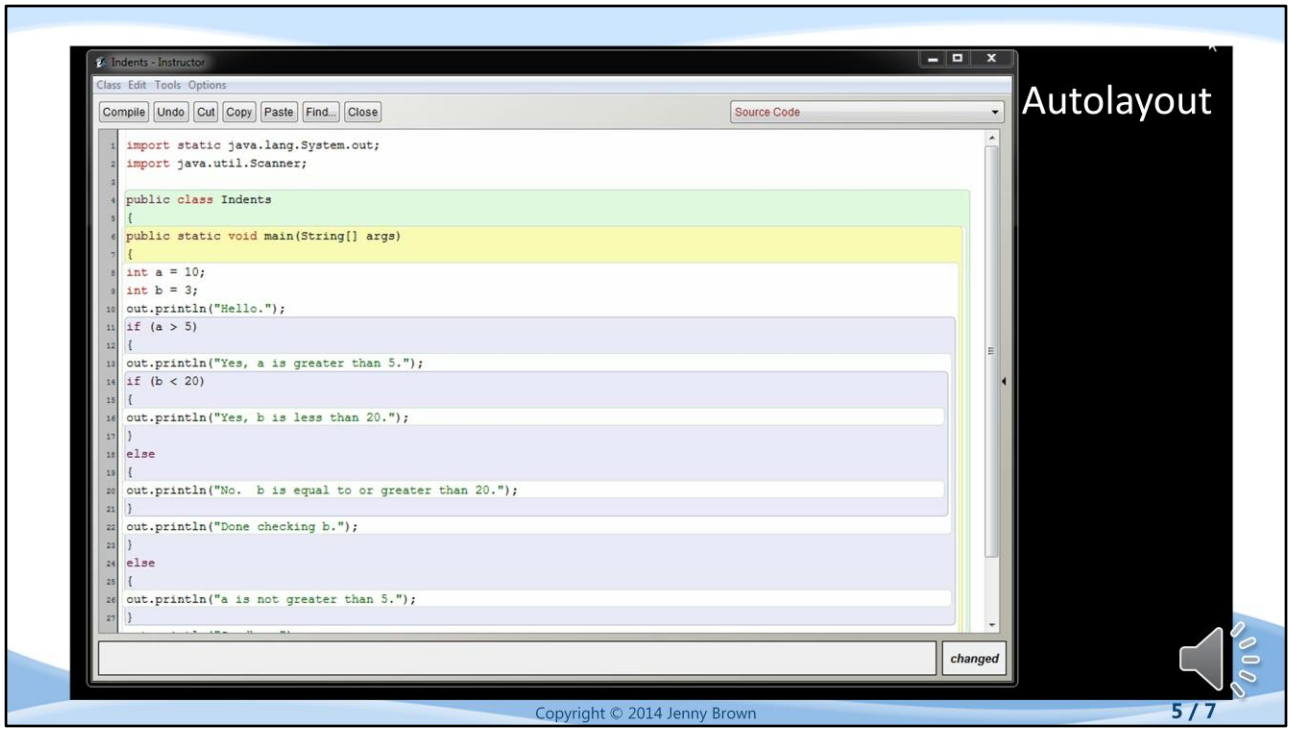

Here's an example of an if statement inside another if statement, as colored in BlueJ. I enhanced the colors a bit to make it easier to see.

Notice that BlueJ draws its boxes based on where the curly braces are. Just like in my example, it shows you which statement "owns" the block of code. The main() method owns everything in yellow, from the starting open curly brace at the top, to the bottom close curly brace, and everything in between.

Within that, everything is lined up nicely, here. <click> Once we hit the first if statement, in light purple, then the indent level moves in because of the curly braces, like this. <click> It runs the print statement "Yes a is greater than 5" and then it hits ANOTHER if statement. Once again, the indent level moves in. <click> It comes back out for the closing curly brace, the else, and the open curly brace, and then it goes back in again. <click> Then the else block finishes and it comes back out. <click> The print statement "Done checking b" is back at the same level as the if. You can see the same pattern lower down, with the else block on the light purple if statement. <click>

It can take some time to get used to this process, but once you do, it becomes much easier to understand how your code works, and how the nesting levels work.

BlueJ has a tool that can help you while you're learning, called Auto-layout. Let's take a look.




Here's the same source code example, without any spacing or indents in it.

BlueJ still colors it, but you'll notice it's kind of hard to read without the indents. So there's a tool that BlueJ has to make it very quick to fix this. Go to the edit menu, go to Auto-layout, and click it.

Boom, all of it is lined up, everything in the whole file.

Now if you don't have your curly braces quite right, the auto-layout may still not work perfectly. So you'll have to make sure you get your curly braces matched.

But, the auto-layout can also help you with matching curly braces.

So, this is BlueJ, under Edit, Auto-layout.

## Why learn to indent?

If auto-layout does it, why learn it myself at all?

Helps you read others' code

Helps while you're typing in code

Helps you recognize bugs



6 / 7

So if auto-layout is so powerful, why take the time to learn indenting by hand?

There are a few reasons. One is that it helps you read other people's code. All the examples you find everywhere will use indenting, and as you'll see when you practice, indent makes it just plain easier to read.

Indenting while you're typing in code helps you keep track of your place in the code, so you don't get lost. It also helps you recognize logic bugs that have to do with whether your statement is inside an if block or outside of it, and similar kinds of bugs for code structures you'll learn soon.

Good indenting makes code more pleasant and easier to work with. Indent as best you can while you're typing in code. Then when you're done, use the auto-layout to fine tune it. Notice what moves as a result, to see where you missed a spot. That will help train your eyes to notice what indenting is supposed to be.

## Journal, Remember, and Reflect

When should the next line of code move to the right?

When should the next line of code move back to the left?

Is there anything you are uncertain of, or curious about?





Journal time. When should the next line of code move to the right? When should the next line of code move back to the left? Is there anything you are uncertain of, or curious about?