Skimpy OOP: Introduction to Object-Oriented Programming Using Java

Tim McMichael

Skimpy Contents

[Skimpy OOP: Introduction to Object-Oriented Programming Using Java](index.html)

* [Skimpy Preface: An Excuse for the Inadequacies Within](_skimpy_preface_an_excuse_for_the_inadequacies_within.html)
* [How to Use This Book](_how_to_use_this_book.html)
* [Related Resources](_related_resources.html)
* [0. 0000 - Getting Started](_0000_getting_started.html)
* [1. 0001 - Computers and Coding](_0001_computers_and_coding.html)
* [2. 0010 - Variables](_0010_variables.html)
* [3. 0011 - Methods](_0011_methods.html)
* [4. 0100 - Classes and Objects](_0100_classes_and_objects.html)
* [5. 0101 - Decisions](_0101_decisions.html)
* [**6. 0110 - Loops**](_0110_loops.html)
  + [6.1. Content for this module is under construction. For now, the section headers below direct you to the corresponding chapter in our required textbook so that you can start right away.](_0110_loops.html#_content_for_this_module_is_under_construction_for_now_the_section_headers_below_direct_you_to_the_corresponding_chapter_in_our_required_textbook_so_that_you_can_start_right_away)
  + [6.2. while Loops](_0110_loops.html#_while_loops)
  + [6.3. do-while Loops](_0110_loops.html#_do_while_loops)
  + [6.4. do-while Loops](_0110_loops.html#_do_while_loops_2)
  + [6.5. for Loops](_0110_loops.html#_for_loops)
  + [6.6. OPTIONAL: break and continue Statements](_0110_loops.html#_optional_break_and_continue_statements)
  + [6.7. A Word About Nested Loops](_0110_loops.html#_a_word_about_nested_loops)
  + [6.8. JUST FOR FUN: Recursion](_0110_loops.html#_just_for_fun_recursion)
* [7. 0111 - Debugging](_0111_debugging.html)
* [8. 1000 - Arrays](_1000_arrays.html)
* [9. 1001 - Inheritance](_1001_inheritance.html)
* [10. 1010 - Graphical User Interfaces with Swing](_1010_graphical_user_interfaces_with_swing.html)
* [11. Stuff That’s Tacked on the End](_stuff_thats_tacked_on_the_end.html)

6. 0110 - Loops

6.1. Content for this module is under construction. For now, the section headers below direct you to the corresponding chapter in our required textbook so that you can start right away.

Help Make These Materials Better!

I am actively working to complete and revise this eBook and the accompanying videos. Please consider using the following link to provide feedback and notify me of typos, mistakes, and suggestions for either the eBook or videos:

[CIS150AB Course Materials Feedback (Google Form)](https://forms.gle/4173pZ1yPuNX7pku6)

*Source code examples from this chapter and associated videos are available on*[*GitHub*](https://github.com/timmcmichael/EMCCTimFiles/tree/4bf0da6df6f4fe3e3a0ccd477b4455df400cffb6/OOP%20with%20Java%20(CIS150AB)/06%20Loops)*.*

A computer is designed to execute a series of instructions, in order, very quickly. Now that we understand how to use Boolean expressions to control the flow of our programs, we can use that concept for a programming structure that really unlocks the power of the computer: *repetition*. Many of the tasks we think of as "computer tasks" are repetitive in nature. Processing data often involves performing the same operation on each piece of information, one after the other, until everything is processed. A computer game has to draw the screen, check for collisions, check what the user is doing with the gamepad, and update the positions of all the objects on the screen, over and over again, until the game ends. When you use a search engine on the web, that search engine has scanned the web one page at a time, indexing what it finds and then moving on to the next page, until it’s scanned all of the pages in its database.

Computers are great for these kinds of tasks: they don’t get bored, they don’t get tired, and they don’t make mistakes—​other than the mistakes we make when programming them!

To create the kind of repetition that leverages this processing power, programmer use *loops*. A loop is a structure that repeats a block of code as long as a certain condition is true. Each time the loop repeats, it’s called an *iteration*.

Video Transcript

often when we're developing a computer program we realize that there is a task or a set of instructions that we need to execute over and over again and in fact this is a big part of what makes computers so powerful is that they can take a task whether it's trivial or something really complex they can do it pretty quickly often very very quickly and then they can just do it over and over again and computers never get tired so they can just do it as many times as they need to do it and uh for example we might be looking at a students percentage in a course and we want to determine what their letter grade is right and we can use for example an if statement to do that we can say well their their grade is higher than 90 so that's an A and so on if you have an entire class of students it's going to be the exact same determination or calculation for every single student right and so what we want is we want to take that little code that figures out their letter grade and we just want to do it with every student in the course to do something like this we're going to use what's called a loop and that's what we're going to take a look at in this video is the basic concepts of what a loop is simply put a loop is a block of code that's going to execute um repetitively based on a Boolean expression and so hopefully you remember that a Boolean expression is something that evaluates to true or false and so for example um the person is over the age of 18 that's either either true or false right that's a Boolean expression uh if the expression evaluates to true then our Loop is going to execute again and if and when the Boolean expression comes back as false the Loop's going to stop we're not going to execute that code again we're just going to keep going on with the rest of our program and so we have uh often we use flowcharts to kind of demonstrate or visualize this kind of thing and so the program execution comes here it's going to take a look at a Boolean expression let's imagine we're writing a simple program that's just going to count from 1 to 10 and so it's going to essentially ask some version of the question are we still counting well if we are still counting it's going to come over here um count up to the next number print out U that number and then it's going to come back ask itself again are we still counting yes we are so it's going to repeat the process again and eventually it's going to be done it's going to be at 10 and it's going to come back false and so the program is just going to continue on with execution from there this kind of a loop um we can talk about this as saying that we're checking the Boolean expression first and then we're executing the code and that's probably the most common type of loop that we're going to um create but there are um other types so for example we can run code first and then after we run the code we can check and see if we want to go back and do it again and um that's less common but it is still uh useful and we we do still often have a need for it uh the difference is or the way we decide which one is better for us is do we have to run the the code at least one time does the code have to run once just to kind of get us started if that's the case then we want to run first and then check um so some examples to make that maybe a little more clear is Imagine we're writing a program and we have to make sure that the user is authenticated that they've they've typed in their password or their pin or or whatever uh we're using to check that they are who they say they are uh so in a check first um approach we're just going to sort of ask the question um is this user unauthenticated right do we not know who this is yet if that's true meaning we haven't authenticated them then we're going to go into the loop body we're going to ask for their password or um their pin number or we're going to scan their Biometrics or whatever and um we can keep doing that until they get it right um in a run first approach then we're going to start by checking their authentication so we're going to start by asking them uh for their password and then we're going to check and see if they are now you know if they're still unauthenticated then we're going to go back up and um and execute that

again in order to talk about these Loops we're going to need a little bit of terminology here so that we can get on the same page and so um we're going to refer to a single execution of that block of code as an iteration right we're going to say that um if we're counting from 1 to 10 uh each number is one iteration and so we are iterating 10 times and we can based on that we can kind of categorize Loops in two different ways there's a definite Loop which is where uh before the loop Loop starts we know how many iterations we need okay that's a definite Loop an indefinite Loop would be a situation where at the time we start executing the loop we don't know how many times it's going to need to iterate as an example here uh that's not really computer programming but let's uh think back to last time we were in a high school PE class and uh the the PE teacher takes us out to the track if the PE teacher says run three laps that is a definite Loop because before we start running those laps we know that we have to go three times right each lap is going to be an iteration an indefinite Loop would be something like the PE teacher telling you to run a lap and keep running laps until your heart rate is 160 or something like that um so you're going to run one lap and as you start you don't know well I'm I might be done after this one iteration uh or I might have to go five times I don't know so you're going to run a lap at the end of the lap you're going to check your heart rate and then you're going to decide if you keep going so in an indefinite Loop um we might not know well we don't know how many times we're going to have to execute how many iterations we're going to need another term we're going to want to be familiar with is the idea of an infinite Loop and so an infinite Loop is a loop that will execute forever ever um sometimes we do that intentionally and often we do that unintentionally as a bug when we're we're writing our program and um if we go back to our our running track analogy uh an infinite Loop would be something like the teacher says well um keep running laps and I'll tell you when to stop and as you start running uh you see the teacher walk off he quits his job and hops in his car and goes home and now you're stuck just running laps forever so that's a maybe not a realistic example but that is an example of an infinite

Loop a couple other terms we're going to need to know um we use the term control variable to refer to um whatever variable we're using to determine whether or not the loop keeps going and there are a couple different ways we can do that um one real common thing we we often use with the loop is a counter and so a counter is a variable that keeps track of the number of iterations uh We've EX executed so far and often we use that as a control variable so in our track analogy um we would be we would have a counter that is the number of laps we've run and so it starts at zero we run around the track and at the end of the uh first iteration at the end of the lap we're going to add one to that and then we're going to say okay well do we need to run again and we're going to keep um we're going to keep going until we have met the requirement for how many times we need to iterate uh that's an example of a of a counter variable they're not always used to um to control the loop sometimes we have a counter that um is part of maybe like an indefinite Loop and we just need to know when we're done how many times we uh we iterate it um a term that I don't use as often but you do hear it so I want to mention it is a flag variable and so usually when people use that term they're talking about a Boolean variable that they're using um to control their Loop and so for example um they might have Boolean variable that's um you know is unauthenticated to go back to my my password example and so they're going to um set that to or I guess I should use the track example um you know is less than um 150 I don't remember what I said uh so at the end of the lap it's going to check uh your your heart rate and if it's going to be Boolean it's going to be true or false whether or not your um heart rate is above that number and uh that variable once it becomes false is going to be what stops the loop in our password example our authentication example um if we're basing our loop iterations on this um variable that is you know whether or not they're still unauthenticated then once we turn that to to um false then the Loop's going to stop executing so a flag variable is another kind of control variable for a loop and uh as we talk about these control variables we'll go back to the idea of an infinite Loop um one of the main reasons especially beginners but I do it too sometimes um get an infinite Loop accidentally an infinite Loop is when you forget to change that control variable so um if you're in a gym class running around the track I don't think you're going to forget to add one to the number of laps uh but you'd be surprised how easy that is when you're writing code where you forget to say okay we've done another lap and so that number of laps very able stays at zero and that poor kid just keeps running forever right that's a that's an example of an infinite Loop and um I'm guessing that you're going to accidentally do that at some point over the next few weeks so um that's another um place where this idea of a control variable comes in that's given us a basic idea of what Loops are and some vocabulary to talk about loops and so uh that's a good introduction and a good place to stop before we jump into actually writing Some Loops uh using source code

6.2. while Loops

6.2.1. This section is not finished, but in the meantime this content is covered in section 6.2 (page 202) in the textbook.

The first structure Java provides for creating loops is the while loop. The while loop repeats a block of code as long as a certain condition is true. The condition is checked before the block of code is executed, so the block of code might not execute at all if the condition is false the first time it’s checked.

A while loop is exactly like an if statement: a Boolean expression is checked, and if it’s true, the block of code is executed. The only difference is that, after the block of code is executed, the program jumps back to the beginning of the loop and checks the condition again. If the condition is still true, the block of code is executed again.

*Example of a*while*loop*

int count = 0;

**while** (count < 10) {

System.out.println("Count is: " + count);

count++;

}

In this example, the Boolean expression count < 10 is checked; if it’s true, the block of code is executed. The block of code prints out the value of count, then increments count by 1. The program then jumps back to the beginning of the loop and checks the condition again. This process repeats until the expression count < 10 is false.

The result of this loop is that the program prints out the value of count 10 times, starting with 0 and ending with 9. It is a definite loop because, before the loop staarts, it’s already known that the loop will repeat 10 times.

6.2.2. Infinite Loops

One thing to be careful of when using a while loop is the possibility of creating an *infinite loop*. An infinite loop is a loop that repeats forever, because the condition that determines whether the loop should repeat is never false. Infinite loops are a common mistake for new programmers, and they can cause your program to hang or crash.

*Example of an infinite loop*

int count = 0;

**while** (count < 10) {

System.out.println("Count is: " + count);

}

In this example, we’ve forgotten to increment count by 1 after printing it out. count will always be 0, so the condition count < 10 will always be true. The program will print out "Count is: 0" over and over again, forever.

|  |  |
| --- | --- |
|  | Your IDE will have a way to stop the program if it’s stuck in an infinite loop. Often, this is a square button—​essentially the symbol for "stop". In VS Code, you can also kill the terminal window that’s running the program by clicking the trash can icon in the terminal window. Finally, in many cases you can press Ctrl+C to stop the program. |

The simplest infinite loop you can create in Java is:

**while** (**true**) {

*// hey, there's nothing to stop this loop!*

}

This loop will repeat forever, because the condition true is always true. Some programmers use this to start a loop and then use an if statement to break out of the loop when a certain condition is met, but that’s not really an infinite loop—​it just puts the boolean expression that controls the loop in an if statement instead of the while statement. I personally consider it less readable than using a clear condition in the while statement, so I don’t write loops like that.

Video Transcript

in Java we have three different basic types of Loops that we can create we have a while loop a do while loop and a for Loop they're all a little bit different they all uh have certain situations where they work better or worse than the others but we want to know all three right now we're going to take a look at how to create a while loop and a while loop if you remember back to the video on basic Loop Concepts is going to be the type of Loop in Java where we check first and then run our Loop body so that that would be this uh flowchart here that we talked about in the other video it's a good first Loop to learn so I always recommend that students start with this one and let's go ahead and jump into Java and see how this works all right I have a real basic program here for us to start with this one just asks uh the user what is the best community college in Arizona it gets a response from them and if the response is not equal to the correct answer it's just going to Output wrong and so I have a constant up here that uh that we can check against to see if we've got the right answer a couple quick things here just to to remind you uh but when we are comparing strings in Java we can't use equals equals or um not equal the bang equal uh symbol we have to use the do equals method that is different than uh some other programming languages that you might use and it's easy to forget and I forget just like everybody else does but I remembered this time so um let's go ahead and try this program out and just make sure it works oh the other thing I was going to point out is that uh strings in these string comparisons in Java are case sensitive right so if I come in here and I say EMCC in lowercase letters it says that that's wrong if I type in something completely wrong obviously that's wrong and to get it right I have to type in EMCC all capital letters just like that so far so good this isn't the most exciting program in the world but I'm just going to draw attention to the fact that the way our if statement works is it evaluates this Boolean expression if that Boolean expression comes back true it runs the block of code and then it moves on with program execution from there a while loop is actually very similar to that because if you remember our little flowchart it's going to check a Boolean expression just like an if statement if it comes back true it's going to run a block of code just like an if statement uh the difference is it's then going to bounce back and take another look at that Boolean expression so if we just change that word while uh the word if to the word while and execute this again I'm going to get it right the first time so EMCC no change right it just goes on and uh and completes execution of program but if I get it wrong oh so it is printing out the word wrong over and over and over again this is an infinite Loop right this means this is a loop that will go on forever because this Boolean expression is always going to come back as true let's first of all let's stop this there are a couple ways we can stop this in vs code you could just kill the whole terminal down there but um what visual Studio code does while your code is running you've never noticed this before because until now your programs have finished running really quickly but there is a stop button up here so I can just stop that and uh we're we're good to go so let's talk about this infinite Loop for a moment what has happened is that we have created a loop that has a control variable and the control variable is called response and so we're checking against that after you know with really before re iteration and it's never changing so there's never a chance for this to ever become false so to speak so what we really need to do is we need to we need to change that response variable and give the user another chance and so I'm going to just steal that line of code from up

there and I'm going to say that change that to say wrong try again what is the best community college in Arizona now in the back of your mind especially if you're an experienced programmer Arizona uh especially if you're an experienced programmer you might be thinking oh well now we're sort of duplicating this why you know we could use um you you said there was a type of loop that is uh execute first and then check and that might be more appropriate for this that's true and we will get there but our our uh Focus right now is on a while loop so we're going to stick with this for a moment let's go ahead and run this version and again I will get it correct the this time I'll get it correct the first time program finishes executing no problem let's get it Incorrect and see if we still have that infinite Loop wrong try again what's the best community college in Arizona so far so good it's at least stopping to prompt me again I'll get it wrong one more time and now I'll get it correct there we go it seems to have have worked uh there is a little bit of this that's kind of unsatisfying and that is that there's the program just sort of stops and so let's just add a line of code here that says that acknowledges that they got it

right oops system out print line and we'll try one more time just to to get that little satisfying message message at the end so get it wrong and we'll say EMCC there we go that looks good let's talk a little bit about how we did this and so there are three components to a loop that I want you to think about when you are uh creating a loop uh three different things that we did so the first thing we did was we initialized a uh control variable so our control variable in this case is called response then we um evaluated a Boolean

expression and that was the while statement there I talked at length about that and then we updated our control

variable so I want you to remember those three kind of pieces or those three components uh the other thing I want to point out here before we talk about some errors you might make uh is that this would be categorized as an indefinite loop at the time we start iterating we don't know how many times we're going to have to iterate in order to finish this process or finish this Loop right because we don't know how many guesses it's going to take for this person to get it correct they may have gotten it right before the loop even starts and so this if they typed in capital EMCC on line 133 here when we get down to evaluate this Boolean expression it's going to come back false which I know know was slightly confusing because getting the answer correct means that the expression is false um but it comes back false and it just skips okay so when we have this kind of while loop where we we check first and then execute uh the code Block it's possible that we have zero iterations right and that's what happens if we get it right the first time but if they do get it wrong and we start iterating this Loop we don't know they might get it right the next time it may take take them 10 times they may never get it right which would be sad for them to not know the best college in Arizona but uh but it's possible so this is what we mean when we say an indefinite loop at the time we start executing it we don't know how many iterations we are going to run the other thing I want to take a moment to linger on here before we move on is a couple U potential errors here that you might have um one of them well actually we've already made one of them we we had that infinite Loop when we started and I just want to talk about that in these terms now uh we had that control variable response and we never updated it so I do think I kind of mentioned that already I guess I don't need to repeat myself uh but a common mistake that students make uh especially if they're beginners you've kind of finally gotten in the habit of typing semicolon at the end of all your lines here so you may type a semicolon at the end of your while statement right just sort of out of out of habit and um I want you to see what happens when you're run that I'm going to kind of move this up here so we can see a little

better it's the best community college in Arizona I'm going to type that in wrong we are actually stuck in another infinite Loop right and it's kind of an odd one to understand uh Visual Studio incidentally will still let me type down there but that's just because that's the way Visual Studio Works um I I have my visual studio set up to automatically format my code every time I hit save and so it has recognized the formatting in here has recognized that there's something kind of fishy going on you're not supposed to have a semicolon at the end of a while statement so it is assuming that was meant to be on the next line and whites space in in Java doesn't matter so having it on line 17 here is equivalent to having it on the end of line 16 and a semicolon in Java just by itself essentially means do nothing right so we have written a a loop that says as long as the answer is wrong do nothing and so that's what it was doing it was just sitting there doing nothing and it will sit there and do that forever um by the way there is I I think we can also hit contrl C on your keyboard to um to break out of that as well um so that's a real common um mistake and another easy way to get get yourself into an infinite Loop U but that gives us a nice basic example of how a while loop Works let's um uh let's take a look at how to create a uh definite Loop using the while loop structure all right so here we're just going to create a simple Loop that counts from 1 to 10 and to remind us of the three components that we need to have I've I've kind of pre-s slugged those reminders in here so we're going to have a loop that counts from 1 to 10 we're going to need a control variable to keep track of those iterations and it's going to be an integer uh it's going to be a counter if you remember back to that uh the other video and in fact I'm just going to call it counter and I need to set it to an initial value I'm going to set it to zero for now and you're going to see as I you may already have kind of alarm Bells going off why' you set it to zero um we're going to experiment with some things and just see how they work here as we go now we need to check the Boolean expression and again it's just like an if statement except it's while instead so while counter is less than 10 and again some alarm Bells might be going off but while counter is less than 10 that's our Boolean uh expression there it's going to be either true or false then we're going to print that out

and I've um I've learned from my past mistake that I also need to update the control variable now that update needs to happen inside the loop otherwise we'll never get to it right so um let's put update the control variable and what I need to do is I need to say that counter equals counter + one we'll talk about that statement and some different ways we can write it here in a moment but let's test what we have so far

not bad um you can it's hard to see probably on the video but it started counting at zero and it stopped after it printed out nine so it's close it it iterated the correct number of times it's just that it started at the wrong number and ended one number too soon so um we need to talk about how to fix that there are going to be a couple different ways we can do that including some more clever ways than than what I'm going to do now but the the starting point is really easy I can change that that uh initial value for counter and so that's an important thing to remember we don't always want our Loops to start at zero or one or whatever we want we might need to start at some other number that's part of our initialization right I've initialized my control variable to where I want it to start and let's run it again see what we get we're still stopping at nine but we are starting at one so we have solved that problem let's take a look at this Boolean expression while the counter is less than 10 run this code so at the first iteration or the first time we encounter we evaluate this expression one is less than 10 so it comes back true let's run it and we're going to add one onto it now we're equal to two go back up evaluate it again two is less than 10 run the code add one onto it now we're at three and it's going to keep going we won't talk through every single iteration but what happens when we get to 8 and now we add one onto it it's nine okay nine is less than 10 run the code add one to the variable it's now 10 we've printed out the number nine our variable is now 10 so now we got to jump back up here 10 is less than 10 that is not true that's false so it skips down to here and it continues executing from there so you got to think carefully about that Boolean expression that you write there are several different ways to solve this and honestly it probably doesn't matter which one you use um there's a little bit of kind of style to that like some people just tend to write their Loops in some ways and some don't so for example some people don't like to have um don't we'll just say if the counter is while the counter is less than or equal to 10 so let's going to run that again and it worked because when it gets up here um we're at nine we print out nine we add one to it and it's 10 so we come up here now 10 is less than or equal to 10 that comes back true that works um we run it one more time now at the end of this when it's all said and done counter it is actually equal to 11 but it doesn't matter because we're done printing out right so that's that's one common solution to that I think I tend to usually write it that way um but another thing that people do is they just increment they just increase that uh that sort of um boundary by one so they say as long as the counter is less than 11 then we're going to execute this code let's try that and it it works the same way right we say um we print out nine make it 10 comes up here 10's less than 11 but now we um increase it to 11 and we get up here we've turned it up to 11 just like they did in spinal tap and uh 11 is not less than 11 so I know this is getting into the weeds a little bit but it's it's just like with our if statements that logic is so important and it's so easy to make mistakes right and so another common mistake you might make is putting that sign the wrong way I can't tell you how many times I've done that what happens if we make that

mistake well that was weird nothing happened something happened here's what we set the counter to 10 that's our initial value while counter is greater than 11 ah that comes back false so we just skipped down here zero iterations right this is a check first then run kind of a loop if we fail that first check we never even run sometimes that's what we want right like if they got the answer right on that previous example but in this case it's incorrect we really want to run that uh at least one okay I paused the video and I backed us up to where we first started our first version of this Loop that uh incorrectly started at zero and ended at nine and I mentioned that there are some potentially clever solutions to this and it has to do with how we update this control variable so the first one doesn't really fix this but it's worth pointing out that most programmers once they have a little bit of experience aren't going to write the this line of code that way they're probably going to do something like plus equals one right or they're um going which it says in add one onto whatever the previous value was or they're going to use what we call the increment operator which is plus plus okay and um plus plus essentially just adds one onto whatever value is there and so if I run that okay and programmers not liking to type a lot being a common thing theme Among Us most programmers are going to write it this way right this little increment operator is pretty cool because one of the things I can do with it is I could put it right there in my print line statement okay and let's see what that does it still works okay and now here's where we can get start to get fancy I'm just moving my comment because now the line of code that increments or that updates that control variable is actually the system out print line we call this a postfix increment and you know I love to be nerdy about my words here um but what that means is that um when it sees this the runtime goes out and checks memory to see what value is Inc counter and it puts it in there and it executes this line of code and then after it's put it in there it adds one onto it we can also do that increment as a prefix and what that says is add one to it first and then stick the value in where we Bel where it belongs so let's run that and that one um gives us a a true or a correct count from 1 to 10 right so that's a just a clever little thing or a useful little thing sometimes especially I know some people as you get more advanced or you learn more like you're always looking for like just more simple or more efficient ways to do things there's nothing inherently more efficient about how this runs incidentally um but uh but in writing your code some people just like that kind of stuff um it doesn't really matter they are all of functionally um you you you can use whichever method works best for your specific uh loop that you are writing all right so that's um that's a good example of a definite Loop and some uh some different ways we can kind of work with that uh let's take a look at U what's maybe a little more of a it's not going to be Advanced but another way we can use Loops to kind of solve some problems okay I've set up a little uh problem that will require a loop to solve so what we're going to do is we're going to ask the user to type in a number and then we're going to create a multiplication table a simple multiplication table by multiplying that times every number from 20 to 100 right it's enough repetitions that we certainly wouldn't want to do this by hand or anything like that all right so I've got um little placeholders for my my pieces that we need so first I need a control variable this is essentially going to be a counter again right this is a counter control Loop and this time since I'm multiplying numbers from 20 to 100 I might might as well start

there and uh then we're going to evaluate our Boolean expression that's going to be our while statement so while that counter is less than and this time I've learned my lesson less than or equal to 100 all right and we're going

to um do whatever we need to do inside of our Loop and so this is going to be first I need to C so I'm going to multiply the counter times multiplier is what I called the user's input so I need to multiply those two together and um save that in a variable uh again you might could possibly skip the variable for beginners and especially in Java I like to to put this in a variable not just in the print line statement uh so answer equals counter times multiplier and now we're going to print that

out and we're going to I'm going to make it I give it a little bit of kind of user interface here so I'm going to um say counter I'm going to say you know it's going to say 20 \* 7 equals or whatever so uh the the code looks a little bit confusing with all these um pluses and all of that but say multiplier equals that and then the answer see if I got all that correct okay let's give this a shot and see how it

works I'll type in the number

three and we do have an error I have made an infinite Loop and hopefully you are already understanding what the problem is right we have a place fer that says we need to update our control variable but we never did it so let's go ahead and uh we're going to do and I could use I I'm going to put it down here sort of separately um I I just showed a moment ago how we could do it cleverly kind of inside here or something like that but I want to keep them separate because I'm trying to emphasize these three separate points uh that we are going to need to do so let's see if I have fixed it three it's not an inite Loop so that's a good sign I'll scroll up and see if we started at 20 20 \* 3 is 60 21 \* 3 is 63 and so on we should get down to 300 at the bottom and that looks like it's good let's do one more simple one uh to just sort of wrap up uh how you might use these to solve problems okay so there are obviously all kinds of things that you can do with while loops and we're just barely scratching the surface but that does does show the syntax and the basic idea of how it works so we'll go ahead and stop there reach out if you have any questions

6.3. do-while Loops

6.3.1. The videos for this section are not finished, but in the meantime this content is covered in section 6.5 (page 217) in the textbook.

As we’ve seen, a while loop checks the condition before executing the block of code. A do-while loop is similar, but it checks the condition **after** executing the block of code: *run first, then check*. This means that the block of code will always execute at least once. Other than that, a do-while loop is exactly the same as a while loop.

*Example of a*do-while*loop*

int count = 0;

**do** {

System.out.println("Count is: " + count);

count++;

} **while** (count < 10);

This is the same loop as the while loop we looked at earlier, but the condition is checked after the block of code is executed. The while statement is at the end of the loop; the do statement at the beginning indicates the block of code that should iterate.

6.3.2. Choosing Between while and do-while Loops

Both while and do-while loops work well for *indefinite loops* (though they can be used for definite loops as well). There’s nothing in the structure of these loops that requires a counter or other control variable, so they can be used for loops that repeat until a certain condition is met, however many iterations that requires.

In many cases, it doesn’t matter whether you use a while or a do-while loop. You really can use either one to create the same loop. However, in some cases, one might be a better choice than the other. The simple rule of thumb for now is: if you need to guarantee that the block of code will execute at least once, you should use a do-while loop; if you need to check the condition before executing the block of code, you should use a while loop.

Video Transcript

we're continuing our work on different types of Loops in Java and once again there are three different types there's a while loop a Do while Loop and a for Loop and in this video we're going to look at a do while loop we're going to spend this time comparing the differences between while loops and do while Loops so I'm making the assumption that you have already watched the video on while loops and that you've got at least a basic grasp of how those work otherwise I think this video might not make quite as much sense so I would encourage you to stop here and watch that if you haven't done so yet and to Jump Right In uh what we're going to remind you is that when we talk about do while loop Excuse me while loops and the sort of flowchart of how they work the basic process is that we check a Boolean expression if that expression comes back true then we're going to jump down and run a block of code and once we're done with that block of code we will go back to the Boolean expression and start again we'll check and if it comes back true we'll execute or iterate again once that Boolean expression returns false then uh the loop is done and we continue on with the rest of the program so we can describe that as kind of checking first and then running a do while loop in Java gives us the option of doing those in reverse order in other words we're going to run the block of code first and then at the end of that we're going to check a Boolean expression and see if that comes back true then we're going to jump back up and run the block of code again so that may sound like a little bit of a subtle difference uh but it's the difference between um a loop that has to run at least once every time that's what a if we if we don't check until after we've executed after we've iterated one time then we're guaranteed to run the loop at least one time sometimes that's good and sometimes that's not exactly what we want so let's go ahead and jump into Java and take look at how this works we will begin with the same basic example that we used when we started with while loop uh which is this idea of uh you can think of it almost like a password right um trying to get the right uh information typed in in our case we're asking what is the best community college in Arizona and we are expecting the user to type EMCC our basic program just does that um but what we want to modify it to do is keep asking until they get the correct answer the difference is this time we'll use a Doh Loop and you'll see that this is an example where a Doh Loop is probably a better choice so uh we have our I can guess I can just C cut and paste these here save myself some typing uh we have our control variable which is going to be response and I have initialized it here now we're going to um set up our Loop and it's the the next thing we think about is how do we check the Boolean expression the difference here is that we're going to check that after we do our ask of the user so the way we uh write that syntax is we're going to use the word do and then we're going to open a code block and I'm going to end the code block right there

and then add our Boolean expression still using the while keyword so do all of that while uh it's not true that their response do equals I wouldn't don't really want that extra space in there but for clarity as I'm typing I think visual studio will clean that up for me anyway but um while it's not the case that response is equal to the answer in other words as long do this as long as the answer is not correct let me save that and clean it up so this essentially is [Music] our check of our Boolean expression I'll just tack that on at the end there and in this case we've also already taken care of updating our control variable because when we ask for that uh response that is our control variable so now we've got this nice compact Loop remember in the while loop we had to ask them first then then we started our Loop and inside the loop we asked again essentially if they got it wrong so we're saving ourselves just a little bit of work uh let's run this and see if I've done this correctly before I get too excited about this once again we've got this idea that we are case sensitive but I'll get it wrong first uh what is the best community college in Arizona I'll type it in lowercase still wrong let's try uppercase that's correct so our just like that our do Loop now works and so again the idea is we're taking that Boolean expression and we're setting it at the end of that block and then so that the compiler so the runtime knows what's going on um we're sort of prefixing it with the keyword do and we're saying essentially if you if you were to explain this in English or or read it out in English do all of this while this expression is true and so it reads really nicely as well that's really all there is to that one um since that's so easy I thought we would add on uh one little piece here just to to um kind of as we build on to the way we think about loops and things we can do with our Loops I'm going to keep track of how many guesses it takes and so this is essentially a counter right so I'm going to call this num guesses and for now I'll start it out with zero I'm going to set that to zero and at the end when we output um that it that they've got the correct answer I'm going to say um that's correct it only took you X number of tries and I'm also going to slip in one other thing that I'm not going to talk a lot about um but I get tired of of the way Java does uh you know using that concatenation the plus symbol to keep adding on the stuff to our strings so I'm going to use a little trick it's called print F you can go look that up in W3 schools or just Google it um Java print F if you want to see um how that how that works I'm just going to say it only took you and now what I can do is I can put essentially a little placeholder there um that percent sign indicates that it's a print placeholder and then the D represents the data type and here's where Java always has to be slightly confusing uh we think of decimals uh as data types that have a decimal point in them uh when we're formatting strings decimal is a decimal integer of all things and so it's there's a little bit of an inconsistency there um but what this lets us do is it lets us just sort of slip in this placeholder that says well we're going to put in a value here from a variable and then I can use a comma to tack that variable on at the end and uh incidentally I can also also use that uh I can put a percent n and that's a new line character so that it'll go to the next line after it's done I don't use print FS very often here I haven't used it in a while so there's a good chance I messed that up let's go ahead and try it try and then

EMCC okay so my print uh my print F statement worked but it still says that the number of guesses is zero so remember the idea when we have a counter like that in inside the loop we need to increment that uh so I'm going to say num guesses plus plus let's try it again I'll miss it a couple of times here ASA that's clearly not the right answer but now we'll try EMCC and so it says that's correct it only took you three tries and so just another little example of how you can use a counter uh to keep track of how many times it's iterated or how many guesses they've done or whatever so uh that's our first simple example of uh how to use a do Loop instead of a while loop so that was a nice little example of an indefinite Loop using the do while structure right an indefinite meaning we don't know how many times it's going to iterate we don't know how many guesses it's going to take for them to get it right and so with this example we're going to add a couple of wrinkles um most importantly I want to show a definite Loop right um so in our while loop video our definite Loop was just outputting counting from 1 to 10 and so we're going to um we're going to add a little twist here where we're going to make that um instead of going from 1 to 10 we're going to go from uh whatever number of user inputs for the starting point to whatever number they enter as their ending point because I want to show you that you um your control variables and the the what we would call the boundaries of your Loop can be determined by variables right they could they can be user input they could be calculations or whatever um we don't necessarily know as programmers uh what those numbers are going to be but it's still a definite Loop because it uh because at the time the loop starts running it's going to know what those values are we're also going to add one other little piece here just to show you um emphasize some things that you can do with loops that are useful and that is um I want to make make sure that that the user gives me valid numbers right so uh when the user enters a starting number um the number that they enter for their ending number needs to be greater than that right so it our program is not going to be set up to work correctly if they type in five as their starting number and then -200 as their ending number because our program is going to move up right it's going to increment the number in each Loop I could write a loop that would work I could write a program that would work correctly but not this one so what I want to do is after they have given me a starting number my ending number I want to keep asking until they give me a number that is greater than the start okay so uh we call this incidentally we call this input validation uh so Loops are really useful for this I'm I'm validating that the the value the user gives me will work for what we need to do with it uh so I'm going to once again start my do um my my Loop of block I could say here um and I'm going to say while and my Boolean expression is while the ending number is less than or equal to the starting number I need to keep asking right in other words I need to keep going until they give me a number that is greater greater than so um we'll see and this this logic gets confusing even for me and especially when I'm recording and I'm convinced I'm going to make a little mistake so let's see if I do this correctly um while the ending is less than or equal to I called it ending it's end it's less than or equal to start and I think that's the logic I want so if if the um as long as the ending number is is too low we're going to keep

asking I will I the user should not use this I don't need to waste video time making it look pretty so um let's test just that part of it and see if it works because I'm not convinced I did it right so enter the starting number I'll say four now if I put one that doesn't work two that doesn't work Nega 23 that doesn't work how about four that doesn't work either but five should work okay and any number five or greater should work good um this should also work with negative numbers by the way so if I my starting number is -23 and my ending number is -10 that's valid right okay so that user input validation Loop um is working now and I just point that out as a really useful thing to do um users tend to not do the things that we expect this doesn't solve all our problems if the user types in my my example that I always use when I say that you know the golden rule of of programming is that users are stupid and that they do things we don't expect so um we'll put in five here and guarantee if you um ask the user to input an ending number somebody's going to come along and type in purple well purple doesn't work that um causes our program to crash because because we're expecting an integer we don't yet know how to fix that uh which would also be a user uh input validation issue uh we don't know how to fix that yet but um but we can at least make sure they give us a number that is valid okay so now that we do have a u starting and an ending number that are valid we can write our um definite Loop to um to do that so let's say do and what we're going to do is we're going to say system out print line and we're just going to count right so I just need to print out oh I need a counter so I've realized that I need another variable for this so I like to declare all my variables at the top count equals one now um we want to print out the

count okay let's not lose sight of the three things that we need to keep track of with our Loop or three things we need to account for um my control variable now is going to be um while the count

is less than we'll start with less than we're going to to figure out if we want less than less than equal to greater than um less than or equal to and and I I think I'm sort of mentally skipping a step that I should pause and talk about for a moment um that count is going to be the number that we output it really needs to begin with whatever the starting value is right whatever the user typed in for um start so I need to begin by initializing that um that control variable here by saying count is going to be equal to whatever they typed in for start okay now I'm going to print out count and then going to increment count which is where I'm going to update my Boolean expression or excuse me uh update my control variable and then checking the Boolean expression with a do while loop happens at the end and so that's right there so my logic here is that I'm going to um initialize my um my counter to you know before I might have initialized it to one but now now I'm initializing it to whatever the user has typed in as the starting number okay um now I'm going to print that out I'm going to add one to it and if it's still less than end then we're going to keep going we're going to keep Printing and adding and so we may have a what we would call a bounding error here with our um with our Loop the bounds refer to where a loop starts and where a loop ends so let's try this and see if we've got that right I'm just going to put from five to 15 okay so my um beginning number is correct it starts at five and then it adds one each time but we're off by one at the end we shorted it by one and this is just like we saw in that um previous video uh this is because count gets to 14 and then to 15 excuse me uh we print 14 we add one to it and so now we say while 15 is less than 15 and it's not so we need to to probably just add a less than or equal to some people again as a matter of style or preference some people also will solve this that way as long as the count is less than the end + one uh there's not a functional difference in those two ways it's kind of whatever makes sense to you in this case I feel like that's a little more readable we would say it makes a little more sense why we we did that

now let's try 23 to

34 and that looks like it works I'm just going to do one more test of things I'm going to say -4 if I try and go to -10 it shouldn't work but if I go to Five that should work and so it goes -4 and it counts all the way up to five so that one looks like it is correct as well and so we've demonstrated a couple different things here we've used a uh an indefinite loop as input validation to make sure they give us some valid input and then once we have that input we are creating a definite Loop uh but we are doing it based on variables instead of what we would call literal numbers instead of saying while the count is less than 10 that's a good place for us to stop uh I I could of course keep on going with all kinds of different examples but in this case do while Loops are so similar to while loop GRS that I I don't feel like I I want to make the video too long um but of course if you have any questions please go ahead and reach out and ask

6.4. do-while Loops

6.5. for Loops

6.5.1. This section is not finished, but in the meantime this content is covered in section 6.4 (page 214) in the textbook.

Definite loops are really common, especially when we learn about things like *arrays* later on, so Java provides a keyword that allows for a compacy way to create that kind of loop: the for loop. The syntax of a for loop can be a little intimidating for new coders, but it really just combines into one line of code all three of the pieces you need for a loop: initializing a counter, checking the counter, and changing the counter.

for*loop syntax*

**for** (int count = 0; count < 10; count++) {

System.out.println("Count is: " + count);

}

The for loop has three parts, separated by semicolons: 1. Initialize a counter. *Example:* int count = 0. 2. A Boolean expression that determines if the loop should repea. *Example:* count < 10. 3. Change the counter at the end of the iteration. *Example:*: count++.

Once you get the hang of the syntax, the for loop is a really convenient way to create a definite loop.

Video Transcript

we're going to take a look now at for Loops in Java and so for this video I'm assuming that you've already watched the videos on how to do while loops and do while Loops because this video really is just going to be a a comparison to those or or a way to implement those kinds of Loops in a different format a different syntax so if you haven't watched those while and do while videos yet you need to stop here probably and go watch those and then come on back in those other videos we've talked about the idea of a counter control controlled Loop for our examples of like counting from 1 to 10 or counting from whatever number to whatever other number uh where there's a it's a definite Loop where we know the number of times it needs to iterate and we're going to use a counter as our Loop control variable that is a really really common type of loop it turns out that we need to do that very often in computer programming and just as one example that we will get to uh later on in the course with something called arrays if you have kind of a list of data like the the roster of a class I think I used in a previous video and we need to go and calculate the grade for each one of those um that that information is going to be processed using a loop that starts at the top and just goes one at the time at a time down to the bottom of the list or the end of the array so this is a really uh common type of uh code that we have to write because it's so common the developers of java have given us a specific syntax just for for that type of Loop and it's called a for Loop so um the idea of a for Loop really is that it's a just a different syntax to do the same kind of loop we might do uh with a while for example and when I say that to students I always get some students who come up with creative and interesting ways to say well you could you don't have to use it for that you could use a force and they've all this bizarre stuff and that is great and that sounds like a lot of fun to play with but I'm talking mostly to beginners here a for Loop is used when we want to make a counter controlled Loop um that's just really straightforward and really commonly used so I'm not going to get into other interesting stuff we could do the syntax looks like this it's going to be the the keyword for and then parentheses but in that parentheses instead of just having a a Boolean expression like we have in our previous examples we're going to have three separate statements right and um then we're going to have our code block which is basically the body of the loop and those three statements are going to match up with these three kind of Concepts or or um three things we need to be aware of when we are making a loop right we have to initialize a control variable we have to check a Boolean expression and we have to update that control variable well not coincidentally that's what these three statements are there for so let's go ahead and jump into Java and take a look at how this works Works uh when we want to write some code so we have here just a real simple definite Loop that we've implemented with the while keyword and this is the example from uh from that first video on while loops and uh it shows a program that basically just Counts from 1 to 10 okay and so as we've discussed we we initialize a control variable we check a Boolean expression and um then within that Loop body we make sure that we update that control variable the idea with a for loop again is when we have something that's just a straightforward um counter controlled Loop like that we can use a very much uh simplified syntax to do that so what I'm going to do I'm going to just comment out that code up above so that we still have it to look at but I'm going to write the same Loop using this four syntax so I'm going to use the word four and then parentheses and I'll go ahead and add my code block there and in parentheses I'm going to have these three statements so I need to initialize my control variable so I'm going to say int counter I'll point out that I am actually I'm defining my my uh variable there as well by putting the int uh keyword so int counter equals 1 semicolon now what's my Boolean expression counter is less than or equal to 10 semicolon and what is my update to the control variable counter Plus+ and then I'm going to come in here and I'm just going to steal that one of code from right there clean up my typo all right let's give that a shot see if it works and it looks like it has executed correctly so um these three lines of code are equivalent to all of the lines of code and I it's not a really big loop up there but it's equivalent to all the code that we wrote up there using a while loop to do the same thing so a for Loop can look really confusing to beginners and especially if they try and start with a for Loop uh it's just like well what is this and what are all these different things the syntax is much different than other things we've seen so far in Java putting three statements inside of um parentheses like that separated by semicolons it's just sort of unusual but if you just break it down and think about what those three elements are for our Loop and then say okay well this is just an easy way to put those all in one line of code then you're covered now just like with any other counter controlled loop I can manipulate these values however I want if I want a loop that goes from 20 to 200 counting by five and I could change it to that okay so I'm going initialize my value my um my control variable B to start at 20 I'm going to keep I'm going to check that it is less than 200 and I'm going to keep iterating as long as it is and then I'm going to add five to it each iteration let's go ahead and test that and see okay again not the most exciting uh or interesting Loop here uh but the point is these all function the same way within a a for Loop statement as they would function independently if we wrote it all out as a while loop so it really is just a shorthand for the same kind of loop that you might write and on that note there's nothing you can do with a for loop at least for now that you can't also do with a while or a do while loop so in that sense it's almost as if a four Loop is kind of optional and if you're really struggling to get the hang of Loops then yeah stick with while loops and do while Loops until you understand them the for Loop is so useful and so common used in programming that I don't want to tell you it's really optional you can't you can't really go far in programming without running into this and especially when you um when you learn that there are some variations of for Loops uh to help us with with collections you know arrays and array lists and things like that um you really are going to need to use it but if you're just barely treading water with loops you can survive without four Loops for a little bit longer until you understand all of the this a little bit better and that's really all there is to a for Loop it again it can look confusing especially if you're a beginner but it really is a simple idea we're just going to take all those things that we need to do in any counter controlled Loop and we're just going to combine them into one really simple straightforward syntax and there as I said there's nothing you can do in one Loop type that you can't do in the others uh it really is just a matter of what fits best for what you're trying to do at the time my rule of thumb to to students and again this is just a rule of thumb it's just a guideline or a starting point but I I tell students if they're not sure decide if it's a definite Loop or an indefinite Loop um indefinite Loops often work really well with while loops and do while loops and definite Loops often work really well with four Loops but again you can do any kind of loop with any kind of loop structure that you want uh for the most part and um you'll be fine my advice to students is that I I really want you to know how to do all three so in my assignments I I will sometimes call out and say well I want a for Loop that does this or I want a while loop that does that because I want to I want you to demonstrate that you've learned them all but really it's just a matter of what fits the situation best and what you think is going what makes sense to you um for that particular problem so um if you want to practice more with for loops and here I've only given you this one example you just take some while loop examples examples and you you convert them and you you rewrite them as for loops and and you'll be getting practice with that syntax I'm also going to recommend um something like W3 schools has a really nice section on for loops and it has um kind of a a virtualized environment there where you can you can try the syntax and you can mess around with it so that's also a good place to go um as is always the case just re reach out if you have any questions

6.6. OPTIONAL: break and continue Statements

I believe that while, do-while, and for loops written with clear Boolean expressions are the most readable loops, and any loop a coder will need in their career can be written with those structures. A well-written loop will execute the block of code as many times as necessary, and then stop when the condition is false, without any additional help from the programmer. However, Java provides two statements that can be used to control the flow of a loop: the break statement and the continue statement.

Since they aren’t necessary for writing loops, I consider them optional: none of my assignments or quiz questions will require you to know them.

The break statement is used to exit a loop early. When the break statement is executed, the loop stops, and the program continues with the next statement after the loop; think of it as a return statement for a loop (except that it can’t pass a value anywhere). Some programmers use break when they need to get out of a loop before the controlling condition is false. My own opinion is that this is a sign the controlling condition should be rethought, but because you’re likely to see break in other people’s code, I think it’s important to know about it.

The continue statement is used to skip the rest of the block of code in a loop and jump back to the beginning of the loop. When the continue statement is executed, the program stops executing the loop’s block of code, and jumps to the Boolean expression that controls the loop to see if it should run again.

Video Transcript

all right we need to take just a quick look at the break and continue statements in Java which just give us some ways we can manipulate how a loop executes during program execution I want to make one thing very clear at the beginning that this is completely optional uh you can certainly do all the assignments in this class without knowing break or continue and honestly you could go an entire programming life without ever really needing to use these and I know because that's basically what What I've Done um so if you're a beginner and you're still trying to get your head around how this stuff works I would recommend that you skip this for now and maybe come back to it later uh or if you're just one of those people like just tell me what I need to know to be able to get through the assignments um but I I feel like I need to talk about it because you do see this come up in in code that's out there and uh some people are curious how it works and I I want to spend a little bit of time explaining that so uh if you are interested then you can keep watching now uh in my videos before this um what you've gotten is the way that I think about loops that there's they're controlled by a usually a control variable and there's some logic there that gets evaluated in a Boolean expression and decides whether or not the loop should keep on running and um I think that's kind of maybe the textbook way of explaining how loops work um but there are other ways to think about starting and stopping loops and um those are by using things like break and continue um which I consider to be let's just say somewhat evil now um it doesn't mean that you can't use these Java doesn't care and all of my jokes aside there are plenty of programmers who use break to stop their Loops when they want to stop a loop um I just want to make sure you know that every time you type the word break to get out of a loop somewhere a baby Jedi dies and so I'm not saying you're an evil person if you can live with the fact that you are needlessly killing Jedi babies then that's fine that's on you and I'm going to show you how to do that um but I prefer to write my Loops like a sane person and I don't use break to stop them so with that in mind let's jump in and take a look at how break works all right so let's start out with just some basic explanation of what these are and and um I won't get into why I think they're evil um look the truth is these are not a big deal um but you know what it's my video and I don't like to use break statements and I think they're um problematic for beginning programmers so I can say whatever I want um here's what they are um the break and continue basically just give us some other ways to manipulate how our Loop executes like I mentioned and so the break statement causes the loop Loop to stop without checking the Boolean expression and it doesn't iterate again okay so it essentially um it doesn't I'm always careful the way I say it because it doesn't stop your whole program from running it just stops the loop from running and then it goes on with whatever the rest of the program was uh but there's once you type once you hit a break statement in program execution um you are not going to iterate anymore within that Loop the continue statement is I guess sort of similar but it doesn't stop the loop completely it just stops the current iteration of the loop and moves on to the next iteration right and so in the simple Loops that we're using that's a little bit harder to kind of explain what it does or demonstrate what it does um but um but maybe you want to run half of your iteration so to speak right um and you do a check halfway through and uh you know what whatever algorithm you're writing um and then um you can hit continue and it'll it won't run the rest of the code in the loop block but it will jump back up and start again and you know evaluate the expression just like normal so uh let's see how that looks in code okay so here we have just a real basic count to 10 Loop like we've been looking at in other videos and just a nice innocent little Loop minding its own business not doing anything to hurt anyone much like a baby Jedi um let's go ahead and add and an if statement here that just checks to see if we happen to be on the number five in our count and if we are we're going to break and so break is its own statement it's um it gets its own line of code and a semicolon and um what's going to happen is the Loop's going to execute up until 5 and then we're going to break so let's run this and see what

happens okay so once it got to uh well it printed four um I do my my um increment after that so it it prints four it increments to five now we check and see that it says five so we break and we stop the loop um it still does say all done right so program execution doesn't stop completely we go on with the rest of our program but there's a dead baby Jedi out there now that we have to have to kind of live with on our conscience okay so that's the break statement uh if we change that to continue and run it again we can see how what the difference is between break and continue uh a little bit sort of hard to notice at first um but once it got to five it then stopped that iteration before it printed out uh what the counter is and it jumped back up added one to it and continued so it had the effect of skipping that iteration and if I had written um some code before that that that code would have

executed this is going to look really kind of messy when I do this but it'll at least demonstrate what I mean by that okay so you can see here that it got partway through the iteration and it then it checked and saw oops it's time to continue and now it skipped the rest and just moved on so that's a pretty good simple little demonstration of what continue does and you might think well what's the big deal that seems fine to me and it really is I don't I don't mean to play it up too much but um to me I I just would rather if this Loop needs to let's go back to The Brak statement that continues a little bit weird um if this Loop needs to stop running after it's printed up to four then why don't we just modify the Boolean expression to do that so um partly this is a symptom of this is a really silly example of of how we would do that but I do see students write that code um I think I think where it's more reasonable to use break would be if there's some exceptional circumstance that happens that you're checking for within your Loop that um that really is out of the ordinary and then I feel like that would be an appropriate place to use break and so um I didn't I didn't come up with a fancy example for that but um but that's the break statement and um that's sort of it's basic usage let's take a look at um another kind of um case of using break here all right so this is the start of a program that's going to also just be one of those counters and so um this is where um it becomes a style issue that is um that is this is where my jokes come from this is just not the way that I write Loops it's not the way I think about loops and it really isn't wrong there's um I don't I shouldn't um I shouldn't make too big of a deal with it but like I said it's my video so um what people will do sometimes is they will intentionally write a uh an infinite Loop and then they'll use a break statement to stop it and so we'll get something like this so while true now the keyword True by definition is always going to evaluate to true so this Loop just never stops okay and it'll it'll keep going as I've said before it'll keep going forever unless there's something going on in my program that's you know a memory leak that causes you know resources to run out or something like that um so people will just put while true and then they'll put their logic in here and say well if counter is greater than 11 then break

oops oh and I'm not even think I'm so distraught by the fact that I'm writing this code that I I missed my first instinct uh if the counter is greater than 10 we'll stop there um so again there's really nothing wrong with this I I don't think this is the worst uh kind of crime against humanity that I can imagine um it's not the way that I write Loops but there are a lot of folks that do this um and the only justification I'll give for my my sort of side of this is that when I'm reading code um when I read a while statement or when I read the start of a loop I want to understand how that Loop Works without having to dig into the code right so the way that I would write this um with a for Loop or with a with a control variable there that clearly says we're going to repeat until um as long as we're we're less than 11 um I can glance at that as I'm skimming through code and I know exactly what that Loop does if I see while true then that means I have to Now read the code inside the loop and look for that break statement and then try and figure out the logic of how it stops um again in a simple example like this that's not a big deal and and this again is probably not a very good um not a very egregious crime right this is not that big of a deal um so let's look at something that is a big deal let's look at something that really is um just pure evil okay here we are just our good oldfashioned count to 10 Loop like we've been using um at least we think it's a count count to 10 Loop but then some programmer comes along and does something like

this if counter equals 5 then we're going to say counter [Music] equals

11 I can hardly tell you how frustrating this approach is for the exact reason that I was just talking about that my defense against using break is that I want to be able to look at the at the start of a loop and understand how it works in this case I look at this Loop say oh okay this is I've seen this a gazillion times this is a loop that's going to go from 1 to 10 okay but then the programmer manipulates that counter variable okay and does something funny with it this is really really confusing using for anybody who's reading the code I have no idea why this person did this okay but in the middle of the loop they have ch changed counter to now be 11 which is a value that stops the loop so I guess what they were doing is trying to make a loop that counted to five so um it's just needlessly confusing and I don't know I the only thing I can think is that maybe while people are developing something something they they sort of you know weird things happen in the heat of battle when you're writing code and you're trying to fix things and debug things and you you're trying different things to make it work and then somebody tries this and all of a sudden it works and so they just leave it in um I just think that this is um this is needlessly confusing you shouldn't really manipulate your counter your Loop control variables in ways that are unexpected right if you do then I sure better see a nice detailed comment or comment block that explains um what you're doing

exactly uh so I'm not going to go on with more of these because I just will have a heart attack and die or something like that but um the idea here is that there are other ways you can control your Loops other than the the ways that I've explained them um some of them are really not a big deal if you want to write an infinite Loop and then use a Brak statement to stop it I think as long as that read clearly in your code then that's not a big deal at all um but I would be real cautious if you find yourself having to do weird things weird if statements to make your Loop stop when you want it to stop I would at that point I would encourage you to just take a nice deep breath um maybe walk away get a cup of coffee or something um and come back and think about what am I trying to do with this Loop where do I want it to start where do I want it to end what is uh the condition that keeps it going and see if there's a way to just rewrite it a little bit and make it uh make it work um without any um confusing code in the middle of it so uh if you have any questions or if you want to send me emails with um infuriating examples of this that you can come up with then of course Reach Out

6.7. A Word About Nested Loops

You can put a loop inside another loop, and that’s called a *nested loop*. They are useful in some situations, and studying them can improve your ability think through loop-based algorithms.

However, they are beyond the scope of this course, which focuses on the fundamentals of object-oriented programming.

If you want to explore them, the textbook addresses nested loops in section 6.6 (page 220), and there are many great resources available on the web and YouTube.

6.8. JUST FOR FUN: Recursion

Content labelled as **Just for Fun** is not required for the course, but is included for students who are interested in learning more about the topic. If you’re struggling to learn the material in this course, please skip this—​maybe you can return when you have more time.

6.8.1. I plan to record a brief video on recursion, but it’s not a high priority. This topic is not covered in the textbook. IF your *really* can’t wait, search for the topic on the internet.

← Previous: [0101 - Decisions](_0101_decisions.html) | ↑ Up: [Skimpy OOP: Introduction to Object-Oriented Programming Using Java](index.html) | Next: [0111 - Debugging](_0111_debugging.html) →

Last updated 2025-02-13 19:39:09 -0700