## **1000 - Arrays**

### **Content coming soon!**

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)

### **What’s the Point?**

* Understand the purpose of arrays
* Create and use arrays
* Write loops to traverse arrays

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)/08%20Arrays).

### **.1. Arrays and Indexes**

The variables we’ve used to this point can only store one piece of data at a time, and sometimes we need to store more than that. For example, we might need to store a list of items to buy at the grocery store. I could create a variable for each item, but that would be a lot of variables to keep track of. Arrays give us a way to store multiple pieces of data in a single collection.

Each individual value in an array can be reffered to as an element. Each element in an array has a unique index that tells us where it is in the array. The number of elements in an array is called the length of the array. The first element in an array has an index of 0, the second element has an index of 1, and so on. The last element in an array is always the length of the array minus one. If our grocery list has 5 items, the indexes of the items would be 0, 1, 2, 3, and 4.

Time to Watch!

Introduction to Arrays

### **.2. Defining and Using Arrays**

In Java, arrays are objects, so they have to be created with the new keyword, and array identifiers use square brackets [] to indicate that they are arrays.

The following example shows how an array is declared and initialized:

String[] groceryItems = new String[5];

This code creates an array of String objects called groceryItems that can hold 5 elements.

Once we have an array, we can assign values to its elements using the index, which is placed in square brackets after the array name:

groceryItems[0] = "milk";

groceryItems[1] = "eggs";

groceryItems[2] = "bread";

groceryItems[3] = "butter";

groceryItems[4] = "cheese";

Each element in the array is assigned a value, and the index is used to specify which element is being assigned.

Once we have values in an array, we can access them using the index. The element is just like any other variable, so we can use it in expressions, pass it to methods, and so on.

System.out.println("The first item on the list is " + groceryItems[0]);

System.out.println("The last item on the list is " + groceryItems[4]);

groceryItems[2] = "bananas";

System.out.println("Bread has been replaced with " + groceryItems[2]);

When using arrays, we have to be careful to stay within the bounds of the array. If we try to access an element that doesn’t exist, we’ll get an ArrayIndexOutOfBoundsException. This is a runtime exception, so it won’t be caught by the compiler, and it will cause our program to crash. In all arrays, any index less than 0 or greater than or equal to the length of the array is out of bounds. In our grocery list example, any index greater than 4 would be out of bounds.

Time to Watch!

Array Syntax in Java

Files from video:

* Starter code: [BasicSyntax.java](https://github.com/timmcmichael/EMCCTimFiles/blob/main/OOP%20with%20Java%20(CIS150AB)/08%20Arrays/BasicSyntax.java)
* Completed code: [BasicSyntaxFinished.java](https://github.com/timmcmichael/EMCCTimFiles/blob/main/OOP%20with%20Java%20(CIS150AB)/08%20Arrays/BasicSyntaxFinished.java)

### **.3. Traversing Arrays**

Working with individual elements in an array can be useful, but to really see the power of arrays, you need loops. With a loop, we can easily move through an array and perform some task or operation on each element. For example, if we have an array of quiz scores, we could use a loop to add up all the scores and calculate the average.

When we use a loop to go through an array, we are traversing the array. We can use a traversal to put values into an array (which we call populating the array), to modify values in an array, or to read values from an array. We can also use a traversal to search for something within an array.

Most of the time, we’ll use a simple for loop to traverse an array. The syntax for a for traversal is:

*Example of a*for*loop to traverse an array*

int scores[] = {90, 85, 95, 88, 92};

int sum = 0;

for (int i = 0; i < scores.length; i++) {

sum += scores[i];

}

System.out.println("The sum of the scores is " + sum);

In this example, we have an array of quiz scores. We use a for loop to go through each element of the array and add it to the sum variable. Using a variable like sum to accumulate a value is a common pattern in programming, and we refer to such as variable as an accumulator. Notice that we declare and initialize it before the loop, and then we update it inside the loop. If we declare it inside the loop, it will be reset to zero each time the loop runs, and we won’t get the correct sum. It also wouldn’t be accessible outside of the loop due to its scope.

|  |  |
| --- | --- |
|  | At this point, clever students point out that we could get the sum of the scores without using a loop at all, using the Arrays class from the java.util package. Sure, but since we’re learning about loops, that would defeat the point. My assignments for this chapter assess your ability to write loops, so you won’t get any points for code that doesn’t at least try to traverse the array. |

In the example, notice that we use the length property of the array to set the loop condition. That way, this same loop will work for any array of any size. If we hard-coded the size of the array into the loop, we would have to change our code every time we changed the size of the array.

*Example of a hard-coded loop*

int scores[] = {90, 85, 95, 88, 92};

int sum = 0;

for (int i = 0; i < 5; i++) { // Uh oh! What if we add another score?

sum += scores[i];

}

System.out.println("The sum of the scores is " + sum);

Time to Watch!

Loops and Arrays in Java

|  |  |
| --- | --- |
|  | The Lab Assignments in Canvas can be completed using what we’ve covered to this point. You might choose to complete that work now, then move onto the rest of the chapter—​which you’ll need for the Programming Project. |

### **.4. Putting Objects in Arrays**

In Java, an array can hold a primitive type, like an int, or an object. We’ve been using arrays of Strings, which are objects, but students sometimes don’t realize that they can also create arrays of objects they define themselves.

If we were to define a GroceryItem class with fields for the name and the aisle where it’s located, we could create an array of GroceryItem objects.

GroceryItem[] groceryItems = new GroceryItem[5];

groceryItems[0] = new GroceryItem("milk", 4);

groceryItems[1] = new GroceryItem("eggs", 9);

groceryItems[2] = new GroceryItem("bread", 7);

groceryItems[3] = new GroceryItem("butter", 4);

groceryItems[4] = new GroceryItem("cheese", 4);

Putting objects in an array expands the possibilities of what we can do with arrays. Our array can hold multiple objects, and each object can have multiple fields—​this allows arrays to large amounts of data in a single collection.

Time to Watch!

Putting Objects in Arrays [COMING SOON]

### **.5. "For-Each" Loops**

Because array traversal is such a common task, Java provides a special kind of loop that makes it easier to traverse an array. It is officially known as an enhanced for loop, but it is often called a "for-each loop" because it goes through each element in the array. A for-each loop is easy to write, and because it handles index management for us, it is less error-prone than a traditional for loop. However, it is less flexible than a traditional for loop: we can’t use it when we need to know the index of the element we’re working with, or when we need to move through the array in a different order.

|  |  |
| --- | --- |
|  | The for-each loop is optional. You can always use a traditional for loop to traverse an array, and you’ll need to know how to do that for the assignments in this chapter. |

Time to Watch!

"For-Each" Loops in Java [COMING SOON]

# Transcript 1

most of us when we're first learning to write code we begin with things like variables and we learn about functions or methods and uh then we go on to if statements and loops and depending on your instructor you may have classes and objects thrown in there that's the way that I teach it but many other instructors save those uh concepts for a little bit later but all of those are what I would consider to be basic PR programming fundamentals and when we get to arrays we're now talking about what I consider to be the first sort of intermediate level concept arrays are our first data structure so a data structure is just a way that we can organize information in the computer's memory so that we can make it more efficient to work with and arrays aren't the only data structure uh there are lots of them out there and in many cases those other ones are more efficient choices or better choices but arrays are where they all start and it's where we should all start as programmers in my opinion uh because it ties together a lot of those basic fundamental ideas and it also gives us the foundation for all the other data structure work we're going to do uh later on in our programming lives so what is an array again when we first start programming uh we begin dealing with uh variables to store our data and the anal ology I I always use is that a a variable is like a container where we can store one piece of information right so in this case I'm using a can uh to illustrate that and we can give that variable uh a name or an identifier so that we can refer to it in code and whenever we use that identifier in our code um it's going to go out and look in that container and see what value is there and retrieve that for our use now variables are really powerful they they are the again the kind of a building block for everything we're going to do but they do have some important limits right uh a variable Can Only Hold one piece of information at a time so if I've got this container that's meant to store the name of a player and I put the name John Elway in there well if I come along later and now I want to store the name Payton Manning I can't put them in the same variable if I put pton Manning in there John Elway is gone right it only holds one piece of information at a time I can make other variables of course and I can have a separate one for Payton Manning um but that is a limit of uh of the way we can use variables in our code so this is where arrays come in what if we have a whole bunch of information that's related that we want to store um we can think of that as like a collection and so this would be like a six-pack right so we can take a bunch of variables that are all similar to each other and for some reason they um they they kind of belong as a group and we can combine them into one structure that we can work with as a single entity um so like a sixpack and those individual cans in there are still going to have the same limitations of a variable right it can hold beer or it can hold soda but it can't hold both um but now we've got a way to um to start dealing with bigger data sets than just one piece of information and again we're going to use an identifier to refer to that but the identifier is going to refer to the whole collection not just to one part of the collection and so that's usually reflected in how that we name these right so uh a single container might be called player because it holds the name of one player um our array our six-pack uh might be called team because that's got the names of everybody that's on the team um a six-pack analogy is good for just understanding the basic idea but that's not really the way that data is going to be organized in our computer's memory um so there's a more useful way for us to visualize it and that's just this uh this kind of table um of of rectangles that we usually draw to represent our array so we use this little kind of chart to um to visualize this and again we've got these variables that are grouped together and each one can hold a piece of data so in this case each one can hold a name and we're going to refer to each one of those spots in the array as an element and so the rules here are that um each element has to be the same data type right so I can make an array of cans I can make an array of bottles I can make an array of loaves of bread but I can't have an array that has some cans some bottles and some loaves of bread right we have to determine that um the time it's created the other thing we have to determine at the time we create our array is how many elements there are and this is just because the the programming language um is going to you know your computer is going to go out and reserve a chunk of memory for this array and it needs to know how much space it's going to need that's determined by the data type and the number of elements so you have to set up all of that when you first create your array so this array has six elements each um each one is a name we probably going to think of that as like a string right and our identifier here team refers to the whole array not to an individual variable within that array so we need a naming convention we need a way to to refer to those individual uh elements in the array and we use index numbers for that so um these index numbers they're going to start at zero and um they're going to count up from there and what this means is that if I refer to uh the team array index 2 that in this case is Danielle right if I refer to team index 4 that's Javier all right so that's the the way that we can organize this data and refer to the individual elements within it and the total number of elements in our array is what we call the array length so in this case our array holds the names of six different team members that means it is a an array with a length of six there's an important relationship between the index numbers and that length and this is true in almost every single programming language there are a couple oddballs out there but certainly any that you're going to deal with in my classes um the lowest index number in an array is always going to be zero because that's where we start counting and the highest index number in other words the last element in the array is always going to be the length minus one so if the array's length is six the highest or last index is five this is something that is of I don't say confusing to beginners it's not a hard concept but it's really easy to confuse when you're writing code so what this means is that team index zero is valid team index 3 is valid for example team index 5 is valid but if I try and refer to team index 6 that's a problem because there is no such thing in this particular array so we always have to be aware of the length of our array and where those limits are for the the indexing now the next rule when we're working with arrays or the next important concept that we need to understand is that arrays are immutable now the word immutable means you can't change it in the case of arrays we mean that in kind of a specific way so we can change the contents of an array and by that what I mean is that um these individual elements we can change the value that is in those elements if I have Jesse that wants to be part of the team uh and is going to replace Danielle who's injured I can do that okay I can make that kind of a change to the values stored in the array but I can't for example change the data type I can't all of a sudden put in a frog instead of a person for example I I don't know why I picked that um that example but um the data type in this case holds the name of a person so that's what I have to put in there those kinds of changes are okay the kinds of changes that aren't okay are changes to the array structure itself so if I have a player named Isaiah I can't add Isaiah onto this team by just adding another element once I have created the element the excuse me once I've created the array it it has reserved memory for that exact size I can't just decide I now want to make it bigger or smaller for that matter there are some tricks we can do I mean it's not a hopeless situation um what I can do is I can make a new array that's bigger I can make a new array that holds one more person I can copy all the old data into that array I can put the new piece of data into that um that extra element at the end and then I can change where my identifier name team in this case points so that it now refers to this new array and if you're using a managed language like Java or C or something it will then um kind of get rid of that old uh array and free up the data so this can give us the illusion that an array can be changed but it's actually a pretty involved process this is something that we would refer to in programming as an expensive process meaning it takes some computing power to do that and that's just the nature of arrays if you're going to be working with a collection of data that's going to need to change a lot let's like let's say you need to keep track of your little sister's friends and there always there's always all kinds of drama they're fighting with each other and and somebody's kicked out of the group and somebody else is brought back in uh and you're going to have to be constantly changing the size of this array and moving things around then an array probably isn't the best choice you're probably going to want to use use something like a linked list or one of those other data structures that are way beyond our uh our scope here but there are options out there the last thing I want to mention is the close relationship between arrays and Loops an array by itself is really useful and really handy to have around it can save us from making a whole bunch of different variables and we can just make one but what really unlocks their potential is when you start using Loops to process them uh the most common type of loop that certainly that we'll use when you're working with me is what I call a traversal Loop and so a traversal Loop is when we just want to go through the entire array um so we're going to go through and just do something with each element it could mean that we're going to just output the name uh of each person on the team kind of create a roster uh if it's a bunch of numbers we could be going through and and adding them up something like that but a traversal Loop is a loop where we're going to start at zero we're going to end at length minus one and we're going to increment for each iteration doing something to The Loop and uh you'll see how to write these kinds of loops and you will write so many of them that they will become second major so arrays are going to give us our first chance to deal with collections of data or bigger amounts of data which is really what computers are so great at and and what really makes them so powerful at this point you are now ready to go off and start writing some code using arrays which we will take on in a separate video

# Transcript 2

we're going to take a look at how arrays work in the Java programming language now I'm assuming by now that you already understand what arrays are in a general sense so if you don't if you haven't watched it yet go watch the video that is an introduction to arrays the other thing to be aware of is that pretty much every programming language has arrays going back to the earliest programming languages that were ever invented so whatever you're using probably has arrays in some form but it might be a little different than Java Java uses the C based syntax so in other words Java in general is based on the C programming language so there's a good chance that whatever language you're using is close to this but if you're not using Java you should probably go find a resource that is specific to your language with that all that said let's go ahead and jump into vs code and try this out in Java in that first video I used this example Le of a team where we want to keep track of the names of a handful of players you know six or seven players and I've started to write some code to do that without arrays and so uh the first player is John and now if I want to keep track of another name I have to make a new string variable for that so that's player two and that's Marie and if you're just working with a a small number of d a small number of strings in this case just a handful of pieces of data then doing it with individual variables is not that big a deal you know if it's a if it's a pickle ball team or something like that no big deal but what if it's a football team and now we have at least 22 variables that we going to have to keep track of and maybe even more than that if we have Subs things like that very quickly this starts to get unwieldy right it starts to become a real pain to deal with all these separate variables and that's where arrays come in so if I want to declare an array an array is a variable just like in anything else and so if it's an array of strings in this case I'm going to start out by saying string and now I need an identifier so the identifier or the name of this variable should reflect that it's really plural so if we're going to have you know if we're going to go from player to a group of players then players is probably a good name for that um and that we often use that for our our array names as just adding s onto whatever we're keeping track of um but team is also good because it's the idea of a of a group of players as well and it makes it clear in my code that when I refer to team I'm talking about the whole group not just an individual so this creates a string called team that's what we've been doing all along if we want to make this a variable the syntax that we add are these square brackets and we have to start being careful now because we've got curly braces for code blocks we've got parentheses for methods and and now we're adding on these square brackets which are um kind of above your return or enter key on the top right part of your keyboard that uh is always going to represent or refer to an array in the Java programming language so anytime you see square brackets we're talking about an array now what's interesting here is this sort of looks familiar it's a little bit of deja vu because I've got a similar chunk of code right here our main meth method where we begin our sort of entry point for a for a Java program takes as an argument an array of strings the reason for that has to do with command line arguments and we're not going to get into that here but I just want to point out that oh okay we've seen this before now we're starting to to understand the rest of this you know this first line of code we ever wrote that's a big pain in the neck so uh the square brackets in this case indicate that team is going going to be an array of strings and this one this line of code really just declares that variable and doesn't do anything with it it just sort of identifies the name to make this useful uh we really have to specify the size and so we're going to use an assignment statement in Java arrays are objects and so whenever we make a new object in Java we have to use the keyword new and so we're going to say say what do we want we want a new string array that holds and then we've got those brackets again but inside those this second set of brackets that's where we put the length of the array in other words the number of elements we want to have and so I've got on my team here 1 two 3 four five six my length is six that means I'm going to put six right there okay that line of code now is going to declare the name team as pointing to a an array of strings it's going to create that array of strings um big enough for six individual strings this array is not very useful at the moment because it's essentially empty it's it's filled with um with a bunch of you know empty strings essentially you're going to be null pointers in this case uh so to make it useful we got to start putting some some data into there the way that we refer to an individual element in an array is with this kind of subscript uh syn tax I'm going to say team and now my square brackets and inside the square brackets I put the element that I'm referring to and so if we look up here at my code comment we start numbering at zero 0 1 2 3 4 oops five I missed I missed my cursor there but I counted correctly um remember that the last element is always the length of the array minus one so the the length here is six that's what I specified that means my valid index numbers are going to go from zero to to five so let's say team Z team index0 equals John say team index one equals Marie oops I got to put that in quotation marks marine and then before we get too far let's just test that and see how this works so I'm going to say system out crint line team zero system out print line team one fix my typo and let's go ahead and run this so John and Marie this code works but it's turning me into a liar I said the whole point of this was to not have to deal with these variables individually and that's where we're headed with arrays but we have to get there first and so you got to you got to trust me for a few minutes here until we get to that point um but I want to pause to talk about this idea of accessing arrays uh with these index numbers so remember the valid indexes go from Zer to the length minus one which is five in this case and those index numbers can refer to any of those elements if I refer to for example three it's hard to remember this when you're first starting out that's not the third element in the array because we started at zero so 0 1 2 three that's actually the fourth one so if I run that I should get de oh I don't get de because I didn't assign anything to element number three so when we declare an array of objects and a string is an object until we put something there we get something that in Java is called a null pointer so it just says the word null null meaning kind of nothingness right so if I want to refer to element three I've got to assign that now I don't want to spend time assigning all these right now um because what I really want to show is what if we try to do what a lot of students do and say well I want the last one I want the last name in there that's there are six so that must be element six let's try that we have gotten an error it is an exception we call these errors exceptions in Java array index out of bounds exception so the boundaries I keep repeating it because it's important the boundaries for these index numbers are zero and the length minus one if I try and refer to an index outside of that range that's out of bounds the program is going to Well for now it's going to crash we'll learn later in our programming lives how to how to gracefully deal with that but for now we just crash our program and we grind to a halt so it's really important to use these index numbers carefully and to be aware of whether or not you are in bounds as long as I'm in bounds then these array elements are really just in this case string variables whatever your variable your data type is in your array so this team index zero is really the same as player one was in my old version and I can use it the same way so anything I could do with the variable player one I could now do with Team index zero so I can print it out we've seen that I can change it with an assignment statement and that works just fine so I'm trying to drive home the point here that once we have got some data in once we've created this array the individual elements in that array are just variables and there we don't you might not know them yet but there are a whole bunch of of um string related methods and things we can do with strings that um that I can do with with this team index zero team index one any valid index uh so we need to learn that this is really just a way of grouping individual variables together and those individual variables are still usable as variables all right but Tim come on you said the whole point of this was to make our lives easier this is actually worse because player one is probably easier to type than Team Bracket zero bracket right so this is just making things much worse uh let's go ahead and comment out this code and we'll show you the first shortcut you're going to learn with arrays and that is something that we call an initialization list so if we know what data we want to put into our array at the time we are Crea creating it we can just list that data and it will add that in at uh the time the program is executed so I can say I have to start over I've commented out my my my team array so it doesn't exist anymore that's why I've got an error here so let's recreate that string team equals now if I'm going to use an initializer list I don't use the new keyword instead I give it a block of code with curly braces and I can just list my elements in uh with a comma separated list inside of this initializer initialization so Marie it's hard for me to type and talk at the same time but I want to get all these names vessa and I need a semicolon at the end because this is a regular statement and now if I run this code okay John and Marie and that seems to be working fine and now I think before I tried to access the's index and he wasn't in the array yet so let's try that again there we go so this is our first shortcut with an array you can see that now we are starting to potentially save some time and some effort um using this initialization list now a couple things about this first of all you can only use an initialization list when you first create the array so I can't make an array uh called team and then use it for a little while and then go back and use an initialization list uh it is intended to be used it's allowed to be used only when you are creating the array for the first time so that's important to know uh what that means is that you can't make your array with an initialization list unless you know at the time you're making the array what all those values are so in other words for example if you're getting the values from the user they're typing them in you can't use an initialization list right so we'll have some different tricks for that the other thing I wanted to make clear here is that the only reason these are in par excuse me quotation marks is because this array happens to be an array of string objects right so the the quotation marks are related to these being strings if we see an example of an array of a different data type like a double uh we'll make an array of prices and I'm going to use an initialization list um I just write the doubles as I normally would write [Music] them semicolon at the end and that works just fine right but do not put quotation marks around them because now it's going to think they're strings and that doesn't work I cannot put a string into an array of doubles and vice versa once you declare the data type just like you would with any other variable uh you're locked into that and you can only use that kind of data with your array let me go ahead and fix that back to something that compiles now I'm going to say again that I'm going to drive home this point once again that once we have put a value into an array we can deal with that that element just like any other variable and I keep saying that because there seems to be sometimes a little tripping point for beginning programmers where they just make arrays more confusing than they need to be once you've put something in that array I mean you've been working with strings and doubles all along this is no different you're just it's got this funny little syntax with the square brackets so if I want to print out the prices I'm going to change that to prices and now three you might have noticed already is not going to work right because 01 two the highest valid index is two let's go ahe and just print out that first one okay $1.99 makes sense and if I want to manipulate that variable at all that excuse me see now I'm even calling them variables but um that array element I can do that so let's just say um prices zero plus equals 50 cents right inflation we're going to add 50 cents onto our price so try not to make these arrays more complicated than they need to be once something is in the array this is just a double in this case right these are just strings and I can use them the same way that I've been using doubles and strings all along right so I I know I'm repetitive but that really seems to be a stumbling block for students where they get something into an array and all of a sudden their brain hangs up on you know it's an array now it's hard to work with no they're just variables once they're um put into an array it's just like binding all those variables together all right and one last point I don't want to go through and demonstrate them all because it would it would never finish but I can make my arrays out of any uh any data type I want so I could make an array of ints I can make an array of characters I can make an array of um booleans um and then I can make an array of objects and so we've actually already done that with strings but with any object and so in a separate video we will explore that uh that idea but for now this is the basic Syntax for an array and in fact this is a really good place to stop this video because I want you to take arrays in really small um easily digested pieces here so what I would encourage you to do is stop at this point and just go try experimenting with these make some arrays put some data in it um manipulate that data get used to the idea that these really are just variables that are bundled together if you have any questions you can obviously reach out to me and once you have the hang of that you can move on to the next video

# Transcript 3

arrays by themselves can be helpful and they can be useful but what really unlocks their potential is using them in the context of loops and using Loops to process them that's what we're going to talk about now I'm assuming here that you already have a basic understanding of what arrays are and a basic uh understanding of how to create them in Java so if that's not the case you need to stop here and go back and watch those videos and then come back and we'll be ready to go the other thing I want to mention before we jump in is that uh the japa developers have created some tools that will do some of the things we're going to do uh in our lesson for us for example the arrays class does some of this stuff and I want to acknowledge that that's out there and that if you're writing code in a production environment or if you're writing code outside a class that's probably really useful for you unfortunately I need to assess your ability to do things with loops yourself not to call a method that does it for you so I'm going to ask you to write that code yourself rather than using the arrays class or some other class you find that will do some of this work for you um that's why the assignments are worded in a way that require loose and I just think it's important to acknowledge that it's not like I don't know those things exist and it's not like I don't think you would use those in the quote unquote qu real world but I have to see if you can write a loop that deals with an array okay with all that said let's take a look in Visual Studio code how we can do this All right so I've just got a couple arrays here for us to work with one is an array of strings and one is an array of doubles and to start out with let's just take a look at outputting one of these arrays so if I just say output you know print line team let's go ahead and run that and see what we get what you get is probably hard to even see what that is because it's mixed in with all that kind of compiler output but let me show you here in a comment what we got um we got this weird looking um set of information and to make a long story short what it says basically is that this is an array and it's it holds strings and then this is the the address in memory where that is stored it's possibly useful to someone but it's not really useful to us it's certainly not useful to our user um so just printing out that the array itself doesn't really give us something useful if we want to print out what's in the array that might be a little more useful and it's going to be our first example of where we can create what we call a traversal Loop and so a traversal I talked about that in in an earlier video it just means we're going to go through a loop and we're going to do something with every element in it this is going to be done with a for Loop it could be a while loop we're going to write it with a for Loop and it's a for Loop that you're going to write so often that you will soon have it memorized the idea here is that we need a loop that goes through every single valid index in our array the first valid index is always zero and the last valid index is always the length of our array minus one so I'll create a variable called I to be our counter and I'll set it to zero um I is kind of a convention in in programming you can think of it as standing for like an iteration or in the case of an array I like to think of it as representing the index number right uh we're going to continue as long as I is less than the length of the array and this array that we're going to Traverse has six so I'm going to put a six there and I'm going to add one with every iteration so far so good and in fact let's just try it I'll system out line team now we've worked with basic indexing before and so that would print out the um the first element but what it does is it prints that every single time we want that index number to change we really want it to be OD okay now this looks good it actually works I do have one thing I would like to refine on this and that is this Boolean expression that says we're going to uh we're going to keep going as long as we are less than six we may not know the size of our array maybe while the program is running we ask the user how big it needs to be and then we create it um or maybe we come back and change it and we add another name name onto it now we know that an array is immutable from one of those earlier videos so I can't just do this um at runtime I can't just add a name on but I can when I'm coding I may come in and add this and I may forget now to go back and find all of these traversals and remember to change that from a six to a seven and then what happens 20 minutes later when I need to add another name now I got to go change them to eight so what we do instead is we refer to the length of the array and there is a property that will do that so if I say the name of the array is team. length in Java that gives us the length of the array no matter what that is and so I don't have to worry about it I just know that that is always going to correctly Traverse a loop and all you ever have to an array excuse me and all you ever have to change on this line of code is the name of the array here so if I want to Traverse through prices and then I want to print out some prices still works okay so that's a nice little tip to make sure that we are um staying in bounds as we say and not using invalid index numbers all right I'm going to take a moment just to identify that as an output traversal Just for future reference so we know that's what it is and let's try another traversal um something that we can't really do with a string but we could do with with numbers maybe we want to know that the the sum of all of the values in that prices array so this um I'm going to call this an accumulation traversal we always use these funny words for this right so in order to get a total from from this I'm going to need a variable that we call an accumulator uh in other words I'm the price the total that we're adding up so it's going to be a double and and I'm going to call it um total price and I'm going to set it to 0.0 to start out and I'm going to use that same traversal code for in I equal z i is less than in this case prices. length and we're going to say i+ plus okay now what I want to do I don't want to print them out every time what I want to do is add the whatever the current price is onto that total price so this is going to be an assignment statement and there going to be a couple ways to write this I could say um total price equals total price plus now I want to pull from prices so prices at the current index okay and then this we don't want to print it here because that's going to show us kind of the work in progress so we can do that just to see um it helps sort of illustrate what's going on so I guess we might as well print that out uh so I'm going to put total equals plus total price all right let's go and run this and see what we get so first it outputs that it outputs the team names which is fine and then it's going to Output every iteration it's going to say 199 and then 4 24 which is the total at that point so it gives us kind of a running total that may not be exactly what we want it might be what we want but in this case I really just want the total at the end um so this idea of an accumulator is not really done or it's not really correct until the loop is finished so we don't want to do anything with it until the loop is all done so I'm going just just going to print out the total price down at the bottom there we go $21.7 I didn't um add those up ahead of time to see if that's right but I'm going to assume that that is correct um because I have the luxury of not turning this in for an assignment you of course should always be um checking that stuff and not assuming that it is correct uh but this is what we can call an accumulation traversal and I'll point out by the way what if we wanted to figure out the average price well the average price now is going to be uh the total price divided by the number of items in that array which is prices. length so if we took total price divided by prices. length we would get an average so that accumulation U Loop or traversal can be really helpful for a lot of different things all right let's see another example of a traversal and this one I don't know if I have a great name for this this is going to be like a modifier traversal and I don't really talk about these in those terms but just since I'm I'm commenting what they all are a traversal that goes through and somehow changes the values inside the array because they're not static the array itself is immutable but the contents within the array are not and that's we're using immutables in a real specific way um but this is just going to be another traversal let's just say that we need to increase the prices by a dollar each you know inflation or whatever prices. length i++ you can already see as promised we're writing this for for Loop um often here so all right so what I want to do now is an assignment statement that changes um a each element of the array so I'm going to say prices at index I equals prices oh I was going to show you up here um we've done this this kind of update um with kind of the the lonand way we could also say total price plus equals price PR sorry for going backwards here but as I was writing the next line of code I was realizing I never circled back and showed that these two lines of code are are equivalent and most people are going to prefer the second one since it's less typing and that's the one I'm going to use here I want to if I want to add a dollar onto each price and I'm just going to say plus equals 1.0 and now how do I know that worked well I'm going to do a traversal Loop that does output to see so I'm going to copy and paste I'm not a big fan of copying and pasting for beginning programmers but it seems appropriate here since it's just that traversal and I'm going to Output the prices um one of the things that's going to happen is my um is my output window down here is going to get to start looking pretty confusing so I'm just going to add a little divider here almost thought I was in Python there for a second just to so I can see the difference in that output there we go so each one of these now should be a dollar more than where they started so 299 325 3.99 and so on so that has worked as well again this is just showing that these variables that we've put into an array aray these values that we've put into an array we can still modify them we can do anything with them that we could do with a regular double or a regular string or whatever that variable type is all right I paused recording so you wouldn't have to watch me type a bunch of stuff but I have added code that's going to ask the user for a name and we're going to use a traversal Loop to see if that name is on our team uh so I've just added the scanner stuff um that should be pretty common or familiar to you by now and I'm asking the user to type that name in so we're going to use a traversal loop we're going to say 4 in I equals Zer same old Loop that we've been using I is less than I'm doing teams now I'm back to teams I said teams but it's team of course and so what we're going to do now is inside this Loop body when we're now going to do an if statement that checks to see if this name is the current element in the uh in the array so we're going to start with an if statement and now strings we have to use dot equals or uh in order to sorry I can't type in and talk at the same time in order to compare two strings we can't just do or we shouldn't just do this for example team I there's a chance that might work depending on having to do with the way Java runtime handles strings but it's not really going to work so we have to do dot equals it's just a quirk of the Java language most other um languages don't really do it that way anymore okay so if the search name is equal to the current name you know the the name at the current index then let's just output system out line let's print out yes that name is on the team okay let's go ahead and run this and see if it works I didn't even think to test my code that I wrote while I was paused uh so let's search for the name DAV because I know that one is in there yes that name is on the team uh now what happen happens if we don't we don't find a name if I put Tim nothing happens we don't get any output and so what students are tempted to do here is add an else statement right okay well else I didn't find the name just copy that no not on the team okay let's run that and see what we get try first of all let's go I'll try Tim no not on the team so not on the team that it repeats it a bunch of times um let's try one that is on the team deid it says no not on the team oops found it didn't not on the team again remember this if statement is happening inside of a loop so it's going to keep going from there and um we have a couple different ways we can handle it um this might be one of those exceptions to my rule about not breaking out of a loop right we might break there um but I still want to be able to or sorry I put the break see I use these so infrequently I didn't even um I didn't even use it right that's not really going to accomplish exactly what we want since we're doing output um but what we might do instead is something like a Boolean variable so we could say Boolean um is found equals and I set it to false and then we're going to say not that code for later um if the name is now we're going to say is found equals true and now after the loop now that that um Boolean variable is essentially an accumulator and it is not accurate it's not complete until the loop is done uh so I'm going to say now I'm going to use that sorry I Frankenstein together my code here we're going to say if is found then print yes we're on the team else print no not on the team so we're not going to print every time in inside the the loop at all we're not going to print any um information in the loop we're just going to do the search and then when we're done we're going to take a look Tim nope not on the team now the test is if this um finds it and doesn't print anything out yes that name is on the team now this might indeed be a place for a break statement right um because once we found the name we don't have to keep looking so I might break there I don't in a case like this a loop is um like this it's pretty basic we don't need to really cut it off but there's nothing wrong with that because Loop's done served its purpose now it's done so uh we could break it there if we want I hit stop before I I made a point that really where I was headed with this break statement is that we would say that this is an indefinite Loop right we don't know how many times it's going to have to iterate before it find something so this is a case where I probably would really write this as a while loop again my aversion to break statements which is completely unfounded and there's no reason for it but I would probably write a while loop that does this it just says essentially while the names aren equal keep going and it's going to iterate every time at the end and um you'd have to have an or clause in there that stops IT from going out of bounds that's kind of beyond the scope of what we're doing here so I won't bother with that at the moment uh but just pointing out there's another way to do that okay um let's take another uh usage case of an array let's say that we don't know uh how how big the team is or how many players are going to be on it we want to get that information from the user so I'm going to comment that out so this is going to work a little bit differently I'm going to just skip a couple lines to set this off visually so now let's say that we're going to Define this declare this team array but we're not going to say how big it is because we don't know how big it is we're going to have to ask that so I'm going to say um int oh sorry I need to prompt first system that out print number of players and this is the real um python way of writing this line of code that I don't really like but it will say int um num players equals I missed next in is the one I wanted and just because I'm in the habit of doing this I'm going to say import that next line just to clear that buffer because if I don't I'm I'm going to forget and I'll be wondering why my code doesn't work I have to clear that buffer after we've gotten a number out of there um so now we've got a value and this is how big we want the team to be now remember we can't use an initializer at this point because we've already made the array so I'm going to say team equals new notice I'm not putting this part I'm not putting the string brackets because we that's part of the definition uh of the Declaration I should say of this array I've already declared it it exists so but I'm writing the rest of that line of code equals new string and now instead of putting a literal number like six in there I'm putting that num players okay that's great but we don't have any names in the array right we just got all those null pointers so what we're going to do is we're going to use a loop uh to Traverse through the array and collect input okay so this will be like an input traversal so same old um same old traversal code here now what am I doing I should just be copying the one right there instead of making a watch me tip and what we're going to do now is we're going to say string I'm not going to write it the python way here which would not use any new variables but I'm going to write it the Java way or the the teaching way um string new name equals nothing for now um and I'm going to prompt the user say input player name and then we'll say new name is going to be equal to next line and now I want to take that new name and put it into the array at index I so team index I equals new name okay I think that's going to do it and then we're immediately doing a traversal after that so we'll be able to see if this works uh and I'm going to start with a low number because I don't want to do a lot of typing so we say three players I'm say John de Javier there we go John deid Javier uh why did it say Javier twice do I have an error or did I have just an extra line of code hanging out there oh this is just a case of me being dumb U that Javier is my input from that line so which I probably could have avoided that confusion by using a print statement instead of a print line statement um let's try that again just because that's a little embarrassing I got to redeem myself here let's put four names John Marie Vanessa I don't remember the order we used before Vanessa read There we go that there was nothing wrong except for my uh my brain not working correctly all right so uh that is a good example of an input traversal and the last thing I want to show here is just a an illustration of how we don't always have to go through these Loops just a straight traversal of 0 to length minus 5 we can have some variations of that that might be useful for us us so uh for our input traversal here let's reverse that let's say that for some reason we want to Output the names in reverse order so that is really the same idea here but we're going to modify this statement and we'll just fumble our way through it because that's what new students do our new programmers do um instead of starting at zero we're going to say well I'm going to start at team. length and by the way as I work some alarm Bells hopefully are going off so let's just switch those around now I've I could run it and demonstrate this but I'm going to tell you that I already have some some problems here um by the way one of the things I'm going to do for the in the interest of speed is I'm going to comment that out and I'm G to comment that out and I'm going to go back to just my pre initialized array here um should have done that I should have paused and done that but okay so we're going to uh we're switching things around so now we're going to start at Team length and we're going to go um down down to zero and we want to go backwards now there's a problem here let's run this and see what happens oh I've got a different problem because I messed something up one it was uncommenting I think oh I was trying to uh trying to declare that pop set the uh the size of that okay let's try that again no what else did I miss just missed on my un editing I will I will leave this stuff in the video just so you can see that I make all the same dumb mistakes that students make it doesn't mean that I am dumb although I might be but uh it just means I made a dumb mistake so we all do that kind of stuff all right so what has happened is the the program ran but I don't get any names out butut and that's because there's more to it than just kind of switching the order of these um we need to start really at the length um minus one remember because that is the um the last valid index and we need to flip this sign because now we're going the other direction and we're going to say as long as I is greater than zero we're going to keep repeating and let's see how that one works Vanessa Javier deid Danielle and Marie and is that our complete list no we're missing John so this is that OBO error off by one we shorted it by one um we do this says stop um keep going as long as we're above zero but zero is a valid index number that's the first one in our array so I need to say as long as we're greater than or equal to zero and I think that's going to give us what we're after here okay so that's uh that's an example of an array that goes through kind of in reverse order um and then I won't actually code these but I'll just point out there's nothing that says we have to go through one at a time maybe we want to increase the price of every other one so that would be I plus = 2 right instead of adding one um so this is kind of your standard what I would call your standard traversal but you don't have to Traverse your array in that kind of standard way you can go through it backwards you can um you go every other one or every fifth one or whatever whatever you need to do to um to suit the purpose that you're you know to suit to to complete the task that you're after this is another example where in this video I'm I'm only having time to just scratch the surface of what we can do with loops in the context of arrays uh but I've shown you the basic idea of a traversal we can output stuff we can input stuff we can um go through the loops in different the arrays in different ways we can search through an array that kind of stuff all the basics that we need and certainly what we need for the assignments that you're going to get We'll be asking you to use these skills but to maybe do them in slightly different ways so at this point I would encourage you to go off and maybe try some of your own try try making some arrays and looping through them just to make sure you've got this down and email me if you have any questions

# Transcript 4

one thing that comes up with arrays that just seems to throw new programmers for a little bit of a loop if you'll pardon that pun is when we ask you to create an array that holds objects and it turns out you've been doing this all along because in Java strings are objects and I think we've you've probably been creating arrays of strings all along but uh but when it comes to designing a class of your own and making an array that holds objects of that type um there's something about that that students sort of have trouble understanding how to interact with those objects within the array so that's what we're going to take a look at here now I'm assuming that you already know what arrays are and a basic understanding of what what they do um that you can use arrays on a basic level and that you can write Loops to do things like traversals if you don't know all of that stuff you need to stop here and go watch those videos um but once you are back from doing that then we'll jump in and write some code I'm going to be using this dog class that I literally just wrote right before I hit record and is probably filled with errors that I will discover as we go but the idea here is I want to create an array that holds dog objects okay it's the same basic concept no matter what kind of object we're working with so I am going to um create create a dog array all dogs there's new dog three okay now no matter how we cut this no matter how we do this we're going to have some typing to do so I can only save you so much typing with our array tricks but let's see kind of both ways we can do this um so this is the the first way we learned to deal with arrays I'm going to say dogs index zero equals new dog now I have to call that Constructor and the Constructor it says we need a name phto and we need to know the dog's age I'm GNA say that phto is four and now dogs one dog one dogs index one is new dog over and over is two and dog two is going to be new dog with the name Spike and Spike is going to be eight okay so I have now created uh an array of dogs and I have put dogs into it and again the idea is that students sometimes think that there's some trick here but look what we did this is no different than saying dog D1 equals new dog P okay page four it's the same thing in fact I'm G to move this up here above so it's kind of out of the way um it's the same thing it's just another object these are just additional instances of the dog class the only difference is we're putting them into an array instead of a dog array uh dog variable so to speak uh let's go ahead and do a quick traversal here to see if that uh if that works correctly and I'm going to pause the video for this and through the magic of video technology I have typed a quick little traversal here and let's run this and see what we get ah so we're back to that idea that when we print just an object we don't get any information from inside the object but we have some getter methods that will let me have access to that stuff I've got get name get age check the smell um I can call those the same way I would call it on a regular variable right so if I wanted to um if I wanted to print out d1's name I would just say D1 do get name well it's going to be the same thing except instead of D1 I'm going to say dogs Index i.get right delete that L code that I don't really need uh but again trying to emphasize this potentially confusing idea that these are just dog objects now that we happen to have put into a an array and we just use this slightly different syntax to manipulate them or to work with them so far so good uh let's take a quick look at how we created this array though uh because like I said there's a lot of typing and there's we can't really do much about that you have to provide the information that's necessary to to instantiate these dogs um but there are a couple other things to be aware of here one of them is if I make that bigger I can assign a dog object and exist exting dog object to our array so I remember I created Fifi up here and just put Fifi into a variable called D1 I can add D1 to the array because it's the right data type right it's a dog and so now when we Traverse we should see Fifi's name on there as well it didn't save us any time but it is something that comes up every once a while you have an object already let's put it into an array but the one you might be wondering about is okay how about that initialization list does that that save us any work it does save us a little bit um but it is potentially confusing the way the syntax Works let's comment out all of that and let's try this with an initialization list so I'm going to say dog dogs equals we've got this list now I'll put the semicolon before I forget it I can put for example an existing dog object I could put Fifi in there right as part of my ization list and there's Fifi but I can also just put new instantiations in there so I can just take a an instantiation statement new dog phto new dog Rover and I'll stop there because I'm running out of space on my screen here as I try and show different things so I can still use that initialization but I can't get around calling the using the word new and calling that dog Constructor to create those dogs and then just one last demonstration of this idea of using these just as very just like you would use any other objects is let's um say we want to have phto play outside and playing outside makes the dog not smell good that's what that method does so it's the same thing we've been doing but just trying to through repetition trying to emphasize that so that's going to be the index is 0 one is phto so index one is phto dot play outside and even Visual Studio code recognizes that this is just a dog object so the kind of intellisense they call it the auto correct feature offers that as a as a potential solution system. out. print line dogs one dot get what I oh I was G I didn't did I make a getter for that oh check smell I called it because they being weird with my names check smell and um check smell returns a string and says phyto stinks if I didn't have him play outside and I run it then he smells clean so I it's just another example of how you know don't let the fact that we've put these objects into an array don't let that throw you off you're still just working with objects the only thing that's different is this array syntax we're going to create an array based on that dog data type that dog class and then we're going to populate it with instantiated dog objects there really isn't anything else to say about the idea of using arrays of objects I guess you know it could never end right I another thing I could show you is how do you make an a class an object that has an array as one of its Fields as one of its instance variables um I'm not going to ask you to do that in um cis1 15ab which is where what I'm recording this video for so I'm not going to get into that here but um the idea is we're just trying to emphasize that these arrays are really just variables or objects that are bundled together into a data structure and if we use that square bracket syntax we can access them and work with them just like any other object or variable if you have any questions about that go ahead and reach out

# Transcript 5

I have one more trick up my sleeve with loops and arrays in Java and that is what I call the for each Loop there are a lot of people who also call it an enhanced for loop I think that's what it was originally called by the Java development team when they added it to the language but I think a lot of people call it a four each Loop it gives us another way to iterate through an array or or other kinds of collections when we know what those are and and it um eliminates the need for us to worry about index numbers or going out of bounds or any of that kind of stuff because it handles it all for us it's optional you anything you can do with a a 4 each Loop you can do with a traditional four loop as well and often it's not very useful to us because it's very limited in how it works but when you do need just a really basic um front to back traversal it gives us a nice easy syntax and we don't have to worry about going out of bounds anything so I'm assuming here that you already know how to work with loops and arrays um and how to make arrays of objects as well and if not you can go view those videos otherwise we will jump in so I've started with two basic arrays here one is an array of the dog class and so you can download that code from canvas it's from a previous video and uh and then I've also got just a double array with some prices in it and we want to have an an example with objects and then an example of an array with Primitives and we're going to start by just a basic traversal as we've been doing in in other videos where we want to Output everything that is in the array so the for each syntax is going to be really simple inside the parentheses here for the force statement I'm going to essentially declare a variable of the same data type as my array so I'm going to do the dogs array first the dog's array has objects from the dog class I'm going to create a dog um variable called D um and then you put a colon the two dots and the name of the array that you want to Traverse and so my array is called dogs and the way this works is that every iteration the variable D is going to be equal to the current dog in the array so in other words the first iteration D is going to be Fifi and then the next iteration it's going to be phto and then Rover and then Spike and then it's going to know that we're done with the array and it's going to stop so there's none of this start at zero end at length minus one kind of stuff it just um works and handles the the array boundaries so let's go system out print line now the difference here when we work with this within the iteration is that the other difference is that I don't need that bracket syntax because inside this Loop body I'm not dealing with an array anymore I'm dealing with a dog object so I just can say dog. get name and I'll make it print out the dog's name and age plus d. getage and let's go ahead and run this and see if it works all right and it does just like that I have created a a traversal that out an output traversal as I called it in another video that just prints out um every every dog in the dog's array and that's all it took was just those those couple of lines of code and again the idea is within that Loop body I'm dealing with just a good oldfashioned dog object or whatever other class we're using for our array and I can do anything with it that I can do with a normal dog so uh I'm going to make I could do this within the same Loop but it's fun to make these and we're trying to learn how to do it so I'm going to just do one again and say four so the way the reason we call it a four each Loop is the way that we would read that out is for each dog D in dogs and then our Loop so that's where the name four each comes from and uh what I'll do this time is I'm going to have the dog have a birthday there's a method called have a birthday that as you could probably guess adds one to their age now I could put this in the same loop I don't it's not like the loop is limited to one line of code so in fact after they have a birthday let's go ahead and print them out again right there and I'm going to print a little divider just so that visually I got confused in one of those other videos I don't want to have that happen again oops I can't type either just a little dividing line so that I can tell which Loop is which in my output so let's try that one all right so um the first time through which is just the output um there are the dogs and their ages then the next the next Loop um the next traversal we have a birthday for each dog and then we print them out again and we can see that their age has has increased by one um for each dog now this is great right this is a nice easy syntax much easier than remembering how to do a a regular for Loop if you're a new programmer so why on Earth would we ever use that old kind of for Loop when we could just use this there are a couple reasons one is that this type of enhanced for Loop or for each Loop can only do a start to end traversal what I would call like a standard traversal it's going to start at index zero it's going to go through each array element in order until it gets to the last element in the array in other words I can't go backwards I can't skip some I can't do anything unusual it's just a front to back traversal that may or may not be a big deal um there are sometimes when we want to move through an array backwards or in some different way the other common reason why we might not use a for each Loop in any um given circumstance is that notice there's no counter variable here there's no this isn't a counter controlled Loop and so that's fine if all we're doing is printing out the um the elements or something like that but sometimes we need that Index right if we're doing a search and I want to search through the array and um and check where an object is or where a value is in that array and then end up with that index number so I can do something else with it later I can't do that here right there is no you know I for example so you know what's funny is I sometimes have students that do something like this they'll create a variable called I and then in here they'll say i++ I guess there's nothing wrong with that um but it seems sort of silly if you need an if you need a a counter like that then just do a regular for Loop so I don't particularly like that but but that is another reason why we might not use a 4 each Loop is because we don't end up with that counter now we still can have an accumulator if we want right I could declare I want to add up the dog's ages I could make a variable um called sum or total age or something and I could add to it within this loop as an accumulator and then after the loop is done I would have the total age of all of the dogs so it that doesn't prevent everything that we might do but but that is a pretty big limitation to um to only be able to go through it in One Direction and to only be able to do it without having a uh a counter it does work essentially the same way if we want to do this with a primitive data type like double or int or any of those those other Primitives for each double P or I'll say double price in prices okay and that's I like the way that reads for each double price in the array prices um again I can just say system out print line and I can print out price and that's going to just print that value out I could also um you know if I could also do some math with it I could um I could say double total I'll do an accumulator like I was just talking about equals 0.0 and I could add on to that I could say total plus equals price and now at the end of that I can print out the total and try that see if I've made any mistakes it looks good um you saw in the other video the other you know we could add a dollar to each one or something like that but that is the basic um syntax and usage for a for each Loop or what some people call an enhanced for for Loop really really useful you can't use it all the time but it's um it's a nice um it's one of those nice to know things not a need to know things right you can you can program without it but um now that you know how easy it is you'll probably want to use this a little bit too I do need to caution you you got to make sure that you use whatever Loop is specified by my assignment if you're turning in an assignment for my class because I do consider four each to be optional and I don't want you using it on assignments until you've shown me that you can use traditional for Loops first so keep that in mind but if you have any questions about for each Loops Reach Out