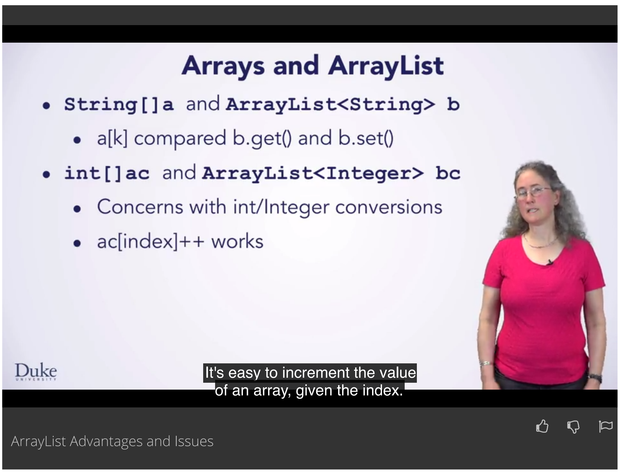
Telling a Random Story

Module Learning Outcomes / Resources 10 min
Introduction 2 min
High-level Design Concepts 5 min
ArrayList 6 min
ArrayList for Unique Words 7 min
ArrayList Advantages and Issues 7 min
Summary 3 min
Programming Exercise: Telling a Random Story 10 min
Practice Quiz: Telling a Random Story 6 questions

Using and Improving GladLibs

Review



ArrayList Advantages and Issues

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript English

0:03
Hi. You've seen ArrayList at work and how useful they are. Arrays are extremely useful too. So we're going to go through a quick walkthrough of code to show where arrays don't work so well. Creating an array is easier in terms of syntax than creating an ArrayList. Far fewer characters to type, for example. It's much easier to access values in an array since a[k] can work to either read from an array location or write to the array location given the index k. In contrast with ArrayList you use .get and .set for reading and writing, respectively. With int values, arrays often have more benefits, in some ways, than ArrayList do. Although most conversions between int and integer happen automatically, occasionally these conversations can lead to hard-to-find bugs if you don't have a thorough understanding of how the int and integer conversation works.

0:56
It's easy to increment the value of an array, given the index. However, in an ArrayList, you have to call .get and .set since the code to simply increment the result returned by .get does not work. However, arrays don't grow, and that's a really large concern. Let's write some code. Now we want to find a number of unique words in a file, but we want to use or at least try to use an array.

1:25
So that's what we're gonna do here. I've started this program, the class is called WordsWithArrays, and the first problem we run in to is we want to read in all the words from a file, but we don't know how many words there are in a file. So we don't know how big to make our array. So we can't really use an array for that. So we'll use a storage resource for that part of the program. So I've already started here. We've got myWords, which is a storage resource.

1:52
We have created this storage resource in our constructor. And then we have read words, which is gonna read all the words from the file and put it into our storage resource, myWords. Node is also going to add them in as lower case, so all of the words have been lower cased.

2:11
We have a method called contains, where we're going to pass in an array of type string and a word and we want to know, is that word in our array?

2:25
So what contains is gonna do is it's going to look through the array and see if the word that we're passing in matches anything, and if it does, it returns true.

2:37
If we go through the whole thing and we don't see it anywhere, we're gonna return false.

2:45
Now, we have number of unique words, and the first thing we've done is we are going to create an array here to store all the words that are unique.

3:00
So you can see that I've started that here. I've gone ahead and putting words as an array of type String. I have to create a new one, so I do that. And then I get to the part about the size, and I don't know how big to make it.

3:14
So I don't know how many unique words they're gonna be. So the only thing I can do is just make it as big as my storage resource. So I made the size, myWords.size. That's the only safe thing to do cuz all the words could be unique.

3:30
Now we're gonna iterate over myWords, and we're gonna check and see for each one is it in words, which is gonna be just the unique words. So, is it already in there? If it's not in there, we found a new unique word and we're gonna put it in there. So, you can see here on this line we add it in. And then we're also keeping track of how many unique words we have because this method is gonna return the number of unique words. And so every time we find a new unique word, we're going to add one to that count.

4:08
The next thing we have is we have a tester method so we can test this out, right down here. So we're just gonna call it and test it out. So let's compile this. It's already compiled, it looks like.

4:20
And so we'll go ahead and run it.

4:24
We have to create our object and then we'll call the tester class.

4:31
And we have to pick up a file, so I'm gonna pick confusius.text. Oh dear, we got an error. So it says down here we got a NullPointerException. Also over here this, is our output here. You can see that it did read in all the words from the file. It says there was 34,582 words read in, but you can see also it got a PointerException and you can see where that exception is. It says in the tester, on line 45, and then in number of unique words, there's a line for that and then contains on line 23. And so that top one is probably where the error is. And if we click on that, it goes to where this error's highlighted here and you can see our errors it contains.

5:21
So what is the problem? Well the problem is we're using this array to put all the unique words but we don't have any in there yet.

5:33
And then we're actually iterating over the whole thing, which is all empty. It's initialized to null. And so we're checking, does a value that's null equal a word? And that's why it crashed. Cuz you can't compare null to a string. So, we need to fix this. So, what we really wanna do is we wanna keep track of how many unique words we have in there. Cuz we just wanna check the unique words that are actually in there that we've actually put in there. And so what we'll have to do in order to fix that is first of all, we'll have to add another parameter so we know how many words we've actually put in there. So I'm gona add a parameter here called number and we actually have to give it a type. So it's gonna be an integer, int number. And then when we iterate, we just wanna iterate over the words we put in there, so we wanna replace list.length. Instead of looking at the whole, gigantic array, we wanna look at just the ones that are already in there. So number tells us how many are currently in there.

6:39
Now we also have to fix where we call contains, which is down here.

6:46
And we have to put a value here. Remember, we're keeping track of how many words we've put in there that are unique words. That is the variable numStored. So we'll pass numStored here.

7:02
Let's compile that and see if that works.

7:05
We got no syntax errors. Let's try and run it.

7:20
And it works. So you can see here we've had a lot of trouble trying to use an array. This problem really should use an ArrayList because here, what's happened is this is a file that has 34,000 words of which only the unique words are 6,558. So that means the array we're using is a size 34,000. But there's only 6,500 unique words. So we have a lot of extra space. That's another reason why an ArrayList would be better for this problem. All right. Happy coding.

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?