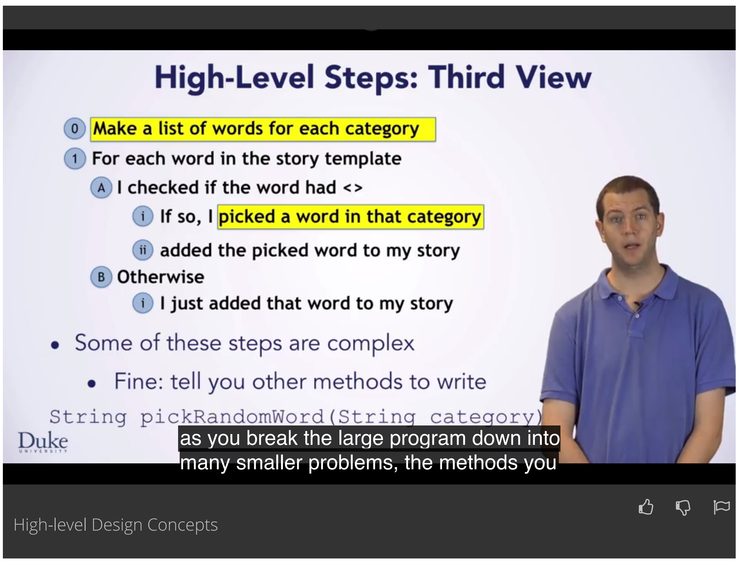

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:03
You're going to think about the design and implementation of a class for generating random stories. This problem is a bit larger than the ones you have solved so far, so you will need to think about the methods that you need and how they work together. The same ideas that you are used to from the seven step process will still guide you through this. You will not only need to think about the algorithm, but also the data involved and how to represent it.

0:24
In Java, thinking about the data and methods guides you through designing a class. As always, you want to work the problem by hand before you do anything else. You could use a story template like this, with pencil and paper, by yourself or with friends. This template is similar to one you've seen before. It uses words and labels to create an interesting story. Let's look at how you might create a story from this template. This story template starts out my name is, then the template requires a name. Remember that a label in angle brackets requires replacement. Here we want a name, so I'll pick my own, Drew.

0:59
After that it goes my job is to, and then we need a verb. I'll let my friend pick a verb. >> Ride.

1:08
>> And a noun. >> Dinosaurs.

1:12
>> If you have ever seen one of these, you know that this job is really adjective. >> Entertaining. >> Because they are so adjective. >> Fluffy. >> Yep, that sounds like a great job to do when I retire, ride fluffy dinosaurs.

1:31
Now, if you started developing the algorithm for this random story program, you might end up with something along these lines. We read each word, saw if it had angle brackets around it, and then if it did, picked a random word from that category. If there are no angle brackets, we just kept the word. But as always you need to be careful as it is easily, easy to mentally gloss over things that happen naturally for you. In particular, we picked random words for each category. But how did we do that? More importantly for this problem, how would you make a computer program do that?

2:06
If I ask you to think of an animal, you can just do it, and it may seem hard to explain an algorithm to think of an animal. As a human, you just know what animals are. You implicitly have a mental list of animals in your head and can just name one of them. It may not be truly random, but maybe you just saw a cat recently or were thinking about your pet dog. But picking some animal is easy. For a computer, however, you need an algorithm and it needs to have data to operate on. The program will need an explicit list of animals to choose from, which could be written into the program source code, or read from a file, or from the internet. So if you think about these steps, there was a step that was implicit for you, but needs to be explicit for the computer. Making a list of animals. More generally, making a list for each template label, not just for animals.

2:55
You should also think about this step, reading each word in a story template. Where did these words come from? This should be some sort of input, like a file or a website. Your program will need to read that file or website, which makes use of familiar classes like file resource, and URL resource.

3:14
You might also notice that some of these steps are a bit complicated. Making a list of words in each category might require more than a few lines of code, though using a file resource or URL resource will help. Picking a random word might also require some planning and programming. It is perfectly fine for your algorithm, and thus your program, to end up with complicated steps. These steps may be names of other methods you will need to write. For example, you might write a method to pick a random word from a category. Suppose the method were named pickRandomWord. If you have this method, the corresponding step in the algorithm is now just one line of code. You just call the method pickRandomWord and it does the work for you. Working through the algorithm development helps you figure out what methods to write. As you write each of these methods, you may in turn find you need yet more methods. Don't let this worry you, as you break the large program down into many smaller problems, the methods you find you need to write will often be simpler than the ones you started with.

4:13
To make the list of words, you will need some variable to hold the data. But how should you store this data? What type are each of these lists of words for template labels? You have seen two types that would work, an array of strings and a storage resource. But neither one is ideal for this problem. Each of these structures has benefits and drawbacks. The StorageResource class is relatively simple to use. Your code can add elements to a StorageResource without knowing how many elements are going to be added. That is, without knowing the number of colors, or nouns, or names that will be added. Accessing StorageResource elements requires using a for loop to iterate over all of them. This will make using an element at random a little tricky to code. On the other hand, string arrays have almost the opposite benefits and drawback. It's simple to choose an element at random, pick a random index less than the size of the array, and return that element, like the element at index two or seven. However, declaring an array variable requires knowing how many elements would be stored.

5:15
That makes arrays not always the right choice. We could use either a StorageResource or a string array to implement this program, but we'll see that a new concept, the ArrayList, combines the best aspects of both arrays and StorageResources. 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?