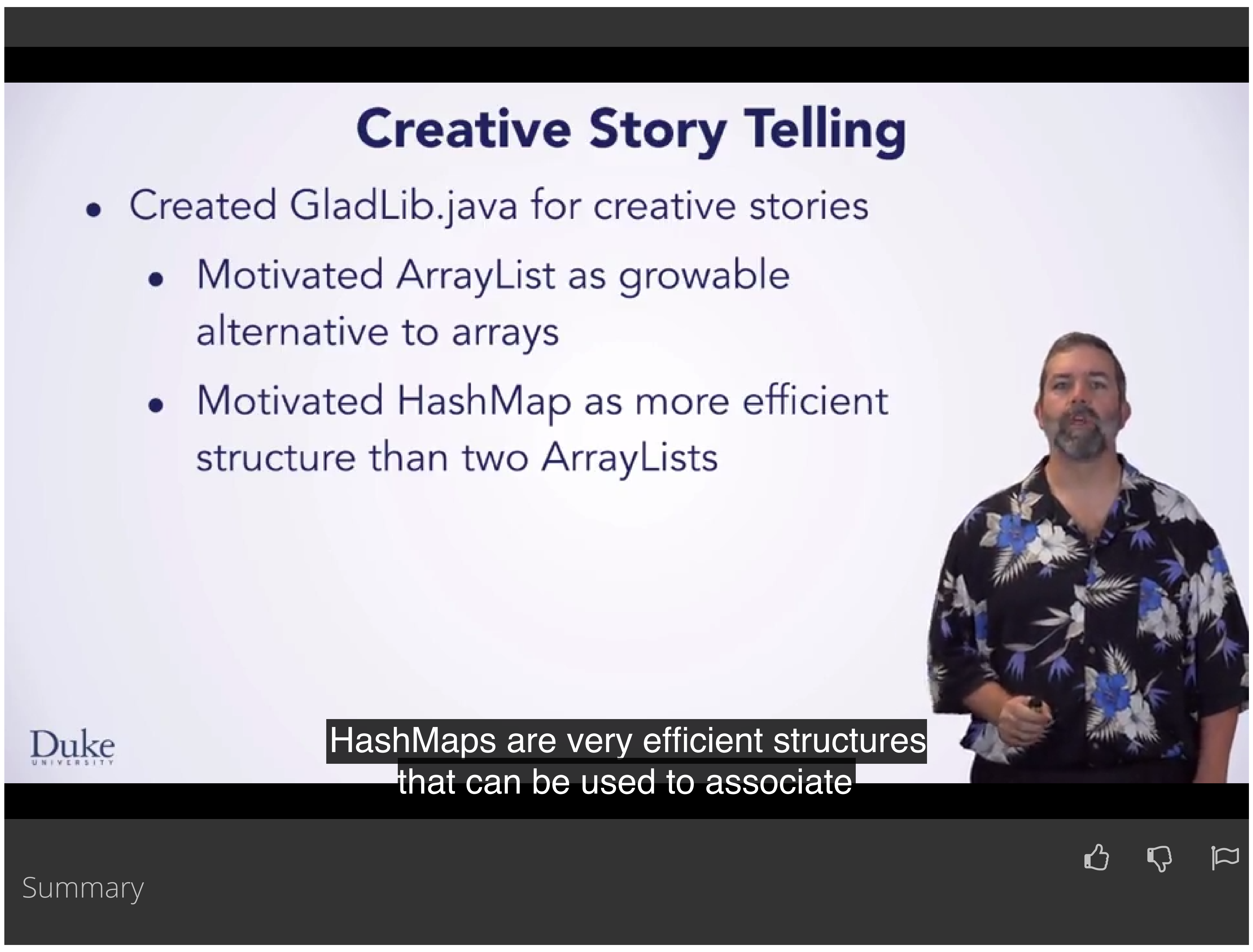

Creative Story Telling

- Created GladLib.java for creative stories
  - Motivated ArrayList as growable alternative to arrays
  - Motivated HashMap as more efficient structure than two ArrayLists

Duke UNIVERSITY
HashMaps are very efficient structures that can be used to associate

Summary

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

Interactive Transcript

Search Transcript English

0:03
Hello, let's summarize what you've had the opportunity to learn studying our GladLib example of creative storytelling. The GladLib.java program read templates from files or URLs to create engaging stories about any topic you chose. We use the program to motivate the need for and study of the ArrayList class. Array lists are like arrays in that they support indexing to individual elements, but an ArrayList object can grow as needed rather than being a fixed size.

0:37
We also use the program to motivate the study of the HashMap class. HashMaps are very efficient structures that can be used to associate keys with values. You saw two examples of this in study of the GladLib class.

0:52
We also use the GladLib class as a small case study to understand that creating code and programs that are extensible is a good idea but requires thinking, planning, and experience.

1:06
We use the GladLib class to study the ArrayList class. An ArrayList object is an indexable collection of elements. ArrayLists store primitive types, so you can store integer objects, but not int values. This means you often need to update ArrayList integer values in two steps. First, getting the value and updating it, and then putting that value back in the map.

1:32
To use the ArrayList class, you must import it from the java.util package. In contrast, you don't need to specify a package for arrays. Useful methods you've seen include add(), to add a new element to the end of the ArrayList. Size(), to determine the number of elements in an ArrayList. Get(), to access an element by its index. Set(), to update an element using its index. And indexOf(), to determine where an element is stored in an ArrayList by index. You can write code to loop over all elements in an ArrayList by either using an ArrayList object as an iterable, or using an Int for loop, starting at zero and looping up to, but not including, the size of the array to access each element by its index.

2:25
We also studied the HashMap class. A HashMap object is a collection of (key,value) pairs. The keys serve as mappings for accessing the values, hence the name HashMap. Both keys and values are objects, so you would use Integer rather than int, just as you did for the ArrayList. Keys are best as immutable objects, like String or Integer. Keys must be unique.

2:52
Values can be any object type, including String or ArrayList as in the examples you saw.

2:58
You import HashMap from java.util just as you do for the ArrayList class. In the examples, you saw the methods put(), to add a key value pair to the map. Size(), to determine the number of pairs or keys in a map. Get() to access a value by it's key. KeySet(), to iterate over all the elements. And ContainsKey(), to determine if a key is in the map.

3:24
Looping over all elements requires an iterable over the key set of the map. You can't access individual elements one at a time by an index as you can with the ArrayList.

3:36
Two different collections with two different sets of strengths. We hope you'll have fun using them.

Downloads

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

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