Telling a Random Story

Using and Improving GladLibs

Review

Following the conventions of writing instance variables, which of the following would be the correct way to add the variable **verbList**?

- public ArrayList<String> verbList;
- private Verb verbList;
- private ArrayList verbList;
- private ArrayList<String> verbList;

Correct

Continue

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

Interactive Transcript

Search Transcript

English

0:03
We'll outline some of the design features of the GladLib class and talk about software design in general.

0:10
Adding a new label, like verb, means modifying the GladLib.java implementation in several different places and requires following a naming convention used in the class.

0:22
You'll need to create a new instance variable for the array list that stores examples of verbs, like run, think, or swim.

0:30
You'll need to initialize the array list using the method initialize from called from the constructor.

0:37
You'll need to get a random verb, when needed, by modifying the code in the method get substitute. You should follow the naming conventions used in the class where the label noun is associated with the instance variable nounList, just as the label country is associated with countryList. This means you should use the name verbList for a field that's the array list that stores strings that are verbs.

1:05
As you modify and extend programs and classes, you'll gain experience with many kinds of programming and design.

1:12
You'll gain experience that will help you make good decisions when programming and create good designs.

1:19
But you'll see that sometimes experience comes from bad judgment or bad designs that, nevertheless, allow you to reason about the trade-offs in doing things in more than one way.

1:30
So you might realize that a choice that works can lead to code that's not easy to maintain.

1:38
Some software designs are called brittle, meaning that the software or design breaks when you try to extend it or use it in ways that is a little bit different than what's initially intended.

1:50
Flexible designs, on the other hand, are better able to cope with changes in the software.

1:58
When you learn about object oriented design, you may come across a principle that says, code should be open for extension but closed for modification, the open/closed principle. The idea is that you should be able to extend software without extensive modifications to the existing code.

2:17
That's more possible with object oriented design ideas that we won't be able to cover in this course, but you can still create designs that are more open than others.

2:27
You'll be able to understand a better design after working with this design and implementation.

2:33
The GladLib class does have some good features.

2:37
It's relatively easy to understand each method, and the code works.

2:43
It is possible to extend the code, as you'll see, even if the extension requires changing the code in several places.

2:51
You'll be able to create a better design after learning some new Java concepts we'll introduce soon, but those concepts will be more clear because you'll have this experience with this code and class, and you'll understand why the new Java features can make the code simpler.

3:09
The new Java features will let you keep changes in one place, rather than being sprinkled across three parts of the class.

3:17
The new features will also minimize the duplicated code that's in the current implementation.

3:22
Have fun making new stories!

Downloads

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

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