Telling a Random Story

Using and Improving GladLibs

Review

Introduction

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

Interactive Transcript

Search Transcript

English

0:03
Now that you've learned about ArrayLists, it is time to work with the Gladlib.java program. You will learn how it works and how to modify it to create stories and new kinds of data. You'll need to understand the programs so that you can modify it. As a programmer or software engineer, sometimes you'll be creating programs from scratch and sometimes you'll be modifying programs. Enhancing them, making them more robust, and more. In this program there are many pieces, many methods each of which you can write yourself. This means you'll be able to understand each method and be able to modify the program in different ways. Since you'll start with a working program, you'll be able to make improvements and additions while testing your program to see that it continues to work correctly. You'll be able to understand class design, method design and the limitations of the GladLib.Java program you start with as you make improvements to the code. As you work to understand these aspects of software design and engineering, you can tell stories about it.

1:03
As you modify the program you'll be creating reusable program components and reusable ideas that you'll be able to use as you develop software experience and expertise.

1:14
We'll take a tour of the program before you begin to make enhancements and improvements. As with all Java classes, a constructor will initialize a GladLib object. The constructor will create the ArrayList instance variables that hold replacements for nouns, colors, and more. It will also create the Java.util.random object used for choosing a replacement at random. The general control flow of the program after initialization will be to read a story template and process each word. If the word is a label, like country or timeframe, indicated by angle brackets, the code will find a replacement at random. After the story is created, the program will print it. Let's look at these pieces in more detail.

2:00
First, let's take a quick look at creating stories from the template you just saw. There is one public method in the class, makeStory. Calling this method will use a template to generate a story as we'll see when we look more closely at the code. In Kenya, a long time ago, nearly 245 decades ago, there live a pink, funny tiger. It so loved to sing and dance, but there was an angry, gigantic lion named Lance that scared it so much.

2:30
In Ecuador, a long time ago, nearly 105 months ago, there lived a jovial, yellow polar bear. It so loved to sing and dance, but there was a furious, angry rabbit named Albert that scared it so much.

2:44
Reading words from the template and printing the story will be code you likely won't need to modify. Calling makeStory will read a template from a file or URL and loop over each word in the template. If the word is a label with angle brackets, it will be replaced.

3:01
Finding labels is a straight use of .indexOf and .substring in the private method processWord. We use these methods to ensure that punctuation or letters before and after the angle brackets are preserved.

3:16
Printing the story will display the final result in the console window of Blue Jay. Or a different programming environment. The private method printOut has a parameter to specify the line width, so you can create a story and use 80 or 40 characters, or any other number.

3:34
You could modify the printOut method to write this story to a file, too, using the edu.duke.fileresource class. Changing the GladLib.java program requires understanding how the ArrayList variables are used. There is one ArrayList for each possible label in a story template like noun or color. These instance variables should be named appropriately so that programmers will be able to easily understand their use in reading and modifying code. As with all instance variables or fields, they'll be created and initialized when the GladLib constructor is called either using new or from within BlueJay when you create an object. The program can use all of the fields when replacing words as part of telling a story.

4:20
The field adjective list will hold replacements for the label adjective. The field nounList will be for nouns and the instance variable colorList will hold colors to be chosen at random. Each field holds replacements for the label that's part of its name. This is a convention in the program not required by Java, but following the convention will make it simpler to create a new instance variable, like verbList, for a new label.

4:46
We'll look at each use of these fields. One is finding a substitute for a label like color. Based on the word that's part of a label like color or noun, the private method getSubstitute will access the appropriate instance variable to find our random replacement. For example if the label is color, then a replacement will be chosen from the field color list. If the label is noun then noun list is used. You can see in the method getSubstitute that adding a new label will require adding a new if statement to access the appropriate ArrayList. A value is chosen at random using a private method, randomFrom. Both randomFrom and getSubstitute are private. They're called as a result of calling the public makeStory method. When getSubstitute calls randomFrom, getSubstitute will always pass one of the instance variables you made such as adjective list etc., as the value for the parameter source. Initializing the ArrayList is straightforward, but you'll need to understand it to create a new field for a new label. All the ArrayLists must be created and initialized when the constructor is called. The constructor will call a private helper method, initializeFrom. The source for the data for colors, nouns, and so on, could be a URL or a file. Calling initializeFrom will result in reading files or URLs to store strings in each ArrayList. If the parameter to initializeFrom begins with HTTP then eventually a URL resource object will be used to read data. Otherwise, a file resource object will be used. You can't see that in the code here because the parameter source is simply passed to the helper method, read it, where the reading code is located.

6:35
Let's summarize how the instance variable for replacing labels are used. You'll need to understand this to enhance the program by adding a new label. For example, to create a label like verb, you'll need a new instance variable. You'll name it appropriately like verbList. You'll need to modify code in two methods with the addition of verbList. You'll modify the method initiazeFrom, a private method called from the constructor.

7:03
You'll modify the code in the method getSubstitute, also a private method, called by the public, makeStory via the private methods fromTemplate and processWord.

7:16
The program documentation should include information like this to help you, the software engineer, make modifications and enhancements. Have fun telling stories and writing code.

Downloads

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

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