Comp 401 - Assignment 4: Interfaces and Text Animation

Date Assigned: Wed Sep 12, 2012

Completion Date: Fri Sep 21, 2012

Early Submission Date: Wed Sep 19, 2012

This assignment will give you experience with defining and using interfaces. You will define interfaces for the scanner and token classes you have written so far. Your scanner will divide words into various commands relevant to the Oz project, and instantiate classes associated with these commands. In addition, the scanner will define an array property that contains an array of token objects. The main class will now animate the scanner by setting different values to the String property of the scanner while it is being displayed by ObjectEditor.

The assignment may be changed in minor ways in response to student questions. So please look for changes in track changes mode before you submit. The changes will also be announced on Piazza.

The following new material is relevant to this assignment.

110 and 401	Interfaces (9/10, 9/12)	PowerPoint	PDF	Interfaces Chapter			lectures.interfaces Package

If you know inheritance, feel free to make use of it in this assignment.

Part 1: Token Interfaces

All of the token classes you have defined so far have an editable string property that stores a substring of the scanned string. Give the same name to this property in all of the classes, define an interface that contains the getter and setter methods of this property, and make all of the token classes implement this interface.

Create additional interfaces for the token classes to follow the constraint that each public method of each of these classes is in some interface implemented by that class.

Part 2: Command Classes

Create a command token class for each of the following command names: “move”, “say”, “rotateLeftArm”, “rotateRightArm”, “repeat”, “define”, “call”, “thread”, “wait”, “proceedall”, “sleep”, “undo”, “redo”. A command token class has the same properties and constructor(s) as the word token class. Thus, you will essentially copy and paste the code of the word token class into each command class, unless you know inheritance.

Part 3: Recognizing Commands in the Scanner Class

Modify the setter method of the scanner to classify word into commands. If the lowercase representation of a word is equal to one of the command names, then create an instance of the associated command token class instead of the word class. Thus, if the scanned string contains the word “MoVE”, you would create an instance of the token class associated with the command “move”. You should use the String equals() (or equalsIgnoreCase()) method to test for equality of two strings. If a word is not one of the predefined commands, then it should be stored in an instance of the word token class, as in the last assignment.

Part 3: Redoing the Scanner Properties

The scanner now has an additional property, which is a readonly stored property of type T[], where T is the interface implemented by all tokens (Please do not name this interface as T!). The getter method of this property returns an array of all token ~~instances~~ objects (instances of your token classes) created while scanning the String property of the scanner. There, should be no empty slots in the array, that is, the length of the array is the number of tokens in the editable scanner property storing the scanned string.

The setter method of the scanner should no longer print the properties of the token objects it instantiates. Instead it should set the value of the variable holding the readonly property. As before, you can assume a limit on the number of tokens in a scanned string. This means that you can create a large ~~intermediate~~ array whose elements are copied into the array returned by the getter method of the readonly property.

Animating Demoing Main Class

As before the main method creates an instance of the scanner class and assigns different values to the editable String property of the scanner. However, it no longer reads input from the user. Instead it displays the scanner object using ObjectEditor and then assigns a series of test strings to the editable property of the scanner. After each assignment, the method should refresh the ObjectEditor window and sleep so that the TAs can see view of the result of each assignment. We discussed this animation approach to demoing in class, and the following code illustrates it: http://www.cs.unc.edu/~dewan/j2h/JavaTeaching/lectures/state_properties/ABMISpreadsheetAnimatingDemoer.java.html. You do not have to select a property or worry about web documents, which this example also illustrates.

I have not tested this kind of animation much with ObjectEditor so it is possible the array will not update correctly in the OE user interface. If that happens, do not display or refersh the scanner using ObjectEditor. Instead, in the animation loop in the main method, before each sleep call, print out the common property of the tokens in the scanner token array.

Extra Credit

1. Instead of printing the errors on the console, store them in a third dependent readonly property of the scanner.

Constraints

1. As always, you can use all of the Java features allowed in the previous assignments and presented in class so far. In addition, you can use equals() and equalsIgnoreCase() String methods.
2. As mentioned above, if you know inheritance, feel free to make use of it in this assignment.
3. As also mentioned above, every public method of an instantiated class must be in some interface implemented by that class. Do not use classes as types of variable or array elements – use interfaces instead. Follow these two constraints in all future assignments also, even if it is not explicitly stated.

Controlling the width of the array elements displayed by ObjectEditor

Some of you will probably make the error property an array and would want to set the component width of the elements of the array.

In general, in addition, you can set the component width of elements of array property P of class C to width by executing the method:

ObjectEditor.setPropertyAttribute(C.class, " P.*", AttributeNames.COMPONENT_WIDTH, width);

For example:

ObjectEditor.setPropertyAttribute(ClassWithArrayProperty.class, "ArrayProperty.*" , AttributeNames.COMPONENT_WIDTH, 800);

makes the component width of the elements of the property ArrayProperty of class ClassWithArrayProperty as 800.

Submission Instructions

- These are the same is in the previous assignment except your document need not contain videos or screenshots. The TAs will run the main method to see the test cases animate.

Good luck!

Formatted: Font: +Body (Calibri), 11 pt, Bold

Formatted: Normal

Formatted: Font: +Body (Calibri), 11 pt, Not Italic, Font color: Auto