CSCI-UA.0101-002: Assignment 7 – Interfaces

Due Monday, December 11th at 11:59 p.m.

Instructions:

- The project directory folder for this assignment is called A7_project_directory_NYUnetID. Rename NYUnetID with your own NYU NetID. For example, my NetID is gp2442, so I would rename my project folder "A7_project_directory_gp2442".
- The project directory contains a project directory containing two subdirectories, namely lib and src. The source files are in src/edu/nyu/cs/NetID. Make sure to rename the subdirectory /NetID to your actual NYU NetID.
- Complete the code according to the instructions in this document.
- Submit a zip file named "A7_complete_NYUnetID" containing your project folder called "A7_project_directory_NYUnetID". Again, NYUnetID should be replaced with your NYU NetID.

Practice with implementing interfaces

In this assignment, you will create several classes that provide different implementations of the same interface, which is given to you. You will also need to write several additional class definitions and other classes from scratch to solve the assignment.

The big idea

Sentences contain words, words contain characters, and the order of these words and characters is important for many human languages. If you were trying to model human written language, you might decide to create a **Sentence** class to represent sentences, a **Word** class to represent words, and a **Character** class to represent characters.

- Since sentences contain words, any Sentence object would have to encapsulate a list of Word objects.
- Since words contain Character objects, any Word object would have to encapsulate a list of Character objects.
- Since both Sentence and Word objects contain sequentially ordered lists of things, you might make them both implement the same SequentiallyOrdered interface to guarantee consistency of behavior.
- Since both Word and Character objects can be stored in ordered lists, you may have them both inherit from a common OrderedThing class that may contain any attributes shared by all ordered things.

Requirements

The SequentiallyOrdered Interface

The following interface code is given to you. The Sentence and Word classes that you will create must implement this interface.

```
java package edu.nyu.cs;
```

import java.util.ArrayList;

```
public interface SequentiallyOrdered {

    public abstract OrderedThing getFirst();
    public abstract OrderedThing getLast();
    public abstract ArrayList<OrderedThing> getSequence();

}
```

The Character Class

You will need to create a class that represents a single character of text. **Character** extends **OrderedThing** because each **Character** object will be stored in an ordered **ArrayList** of **Character** objects in a Word object.

Note: A class named Character already exists in the Java API java.lang package, so your class with the same name hides that one. If you want to refer to that API class (which you shouldn't need to), you'll need to reference it by its full package and class name, such as java.lang.Character in your code.

The Word Class

You will need create a class Word that represents words in a language. **Word** implements the **SequentiallyOrdered** interface, because a word is a sequence of characters. Note that **Word** extends **OrderedThing** because each **Word** object will be stored in an ordered **ArrayList** of **Word** objects in a **Sentence** object.

Instance attributes

The **Word** class should have two instance fields.

- An instance field of type **ArrayList<Character>** which will store a word's character sequence as **Character** objects.
- An instance field of type **int**, representing the **Word**'s position in a **Sentence** in which it is being used (with the first **Word** in a **Sentence** being position 0).

Instance methods

Word should have the following methods:

- getFirst() should return the first **Character** object of the **Word**.
- getLast() should return the last **Character** object of the **Word**.
- getSequence() should return an ArrayList containing all the **Character** objects in the **Word**.
- getPosition() should return the int representing the **Word**'s position in the sentence.

Hint: The **Word** class, which implements the SequentiallyOrdered interface, requires the getFirst() and getLast() methods to return an OrderedThing. Note that a child class can be considered an instance of its parent class; this is polymorphism. So a **Character** object, since it extends OrderedThing, can also be considered an instance of the OrderedThing class.

Constructors

The Word constructor should take two parameters:

- A String parameter and add the individual characters of the String to the **ArrayList<Character>** instance field.

- An int parameter representing the position of the **Word** in a **Sentence** and set the relevant instance field accordingly.

The Sentence Class

You must create a third class, **Sentence**, that represents sentences in a language. **Sentence** implements the SequentiallyOrdered interface, because a sentence is a sequence of words.

Instance attributes

Sentence should have a single instance field of type **ArrayList<Word>** which will store the words of a sentence. This relationship between the **Sentence** and **Word** classes is called composition, because a Sentence is composed of **Word** objects.

Instance methods

Sentence should have the following methods:

- getFirst() should return the first **Word** of the **Sentence**.
- getLast() should return the last **Word** of the **Sentence**.
- getSequence() should return an ArrayList containing all the **Word** objects in the **Sentence**.

Hint: The **Sentence** class, which implements the SequentiallyOrdered interface, requires the getFirst() and getLast() methods to return an OrderedThing. Note that a child class can be considered an instance of its parent class; this is polymorphism. So a **Word** object, if it extends OrderedThing, can also be considered an instance of the OrderedThing class.

Constructors

The Sentence constructor should take a single String parameter representing the **Sentence**, and add each **Word** of the sentence to the **ArrayList<Word>**. You

can split the String into words by using the String split() method in the following way:

```
String[] words = s.split("[^\\w']+")
```

Notice that the split() method will give you an array of String objects, and you will need to go through that array, creating Word objects and adding them to the **ArrayList<Word>**.

The TestSequence Class

Finally, create a test class with a main method that shows how a **Sentence** object can be instantiated with a sentence of your choosing, how each of the methods of the **Sentence** class can be called in a meaningful way, and how each of the methods of the **Word** class can be called on at least one of the **Word** objects encapsulated within the **Sentence** object's **ArrayList<Word>** instance field that you created.