In this assignment, you will gain more experience with defining and instantiating classes. In particular, you will create constructors, and classes with immutable instances. In addition, you will learn how to define and use interfaces.

This assignment is a variation of Part 2 of the previous assignment. In that assignment, each token (number/command/quoted string) was stored in a String or int value (e.g. “10”, “move”), and this value was then used in the readonly property. Now, you will store these token-strings in readonly token objects, which are instances of token classes defined by you. Token-classes will define constructors that take token-strings as arguments. These token objects will be used to calculate the readonly property of the stateful object.

**Token Classes and Interface**

For each kind of token you detected in the previous assignment (number/command/quoted string/sign), create a class whose instances represent the token. All token classes should implement a common interface that defines two readonly properties, StringValue and Description (you are free to rename them). Since the two properties are readonly, the token-objects (instances of the token classes) are immutable. This means they define constructors for storing state. Each of these constructors takes a single argument – the stored token-string.

The values of these properties depend on the token-object. The Description of a token-object is the description you printed in the previous assignment for the corresponding (stored) token-string, without the “:” suffix (e.g. “Number”). TheStringValue of a number token-object is the associated token-string entered by the user to represent the number, the StringValue of a command token-object is the lower case representation of the token-string input by the user, the StringValue of a quote token-object is the quote entered (without any case conversion), and the StringValue of ‘+/-’ is the string “+”/”-”. As we see above, The StringValue of the tokens, in some cases, is different than what you printed in the previous assignment.
To illustrate, for the command token “move”, you will create an instance of a token class that stores the token-string “MoVE”. The Description property of the instance will be the string, “Command”, and the StringValue property will be the string, “move”. Similarly, for the number token “0200”, you will create an instance of a different token class that stores the string “0200”. The Description property of the instance will be the string, “Number” and the StringValue property will be the string “0200”. As mentioned above, the stored values are passed as constructor arguments.

For + and – tokens, you can store the token string as a character rather than a string.

**Modified Class of Assignment 3 Part 2 (the one with properties)**

Define a variation of the stateful class you created in Part 2 of the previous assignment with the same two properties. In this version, you will not compute the value of the dependent readonly property directly from the input tokens. Instead, you will convert each input token (string) into a corresponding token-object and use the values of the properties of these token-objects to compute this property.

Thus the behavior of the class, from the outside, does not change. The only difference is that, rather than building the readonly property directly from token-strings, it first converts each token-string into a token-object, and uses the properties of these objects to compute the readonly property.

To illustrate, suppose the editable property is the string: “Move 020 30”. In the previous assignment, your stateful class broke it up into the token strings: “Move”, “020”, and “30” and used these strings to build the readonly property. Now, you will build the readonly property indirectly from the three token strings. These three token strings will be converted into three token objects, and the properties of the three token objects will be used to build the readonly property. The decision of what values and descriptions to put into the readonly property is taken, not by the scanning class, but by the token classes. For similar reasons, the decision of how to convert a number string such as “020” to a number such as “20” is not taken by the scanner class.

Whether the token objects are stored in a global instance variable (such as an array) depends on whether you stored the token strings in a global variable in assignment 3. If you did not, all you have to do is create a local variable to store all token objects. For each token string you find, you create a token object, use its properties to build the next portion of the dependent readonly property, and then discard the token object by assigning the next token object to the local token variable that stores all token objects.

If you define an error property, then its handling does not change for this assignment.

In addition, define an appropriate interface and constructor for the stateful class.
**Extra Credit**
If you handle + and -, then you will get extra credit for the classes you create for them.

**Main Class**
As in the previous assignment, your main class should create an instance of the stateful class above and use this instance to scan the main argument and print on the console the tokens and their descriptions. You do not have to create an instance of the stateless class or use ObjectEditor. Create screenshots showing the main argument and associated console output for one test.

As always, for the main class, use the naming conventions given to you. Thus put the main class in a package called main, and call it Assignment4.

**Constraints**
1. You can use all of the Java features allowed in the previous assignments and presented in class so far. In addition, you can use the toLowerCase() method provided by a string, which returns the lower case representation of the string.

2. Every class (that is instantiated) must implement an interface that defines all of its public methods, and all object variables should be typed using interfaces rather than classes. Follow this constraint in all future assignments also, even if it is not explicitly stated.

**Submission Instructions**
Same as before. Be sure to follow them carefully, as some of you are submitting only the Java files and not the complete project. Be sure to also show screenshots for all test cases.

Good luck!