These two assignments provide two different implementations of an iterator interface to access scanned tokens. In assignment 3, you do all the scanning at object-initialization time. In assignment 4, you scan as needed. The scanning task is identical to the one in the previous assignment except that you are required to recognize the string token, and do error checking. This assignment will give you further insight into not only iterators, but also scanners, interfaces, and error checking.

**Assignment 3 Task**

As in the previous assignment, you will create a class that scans for words and numbers. However, this class will be different from the one in the previous assignment in the following ways:

1. It will no longer recognize the double quote token, in case you did the extra credit for the previous assignment.
2. It will recognize a new token, called a string, consisting of a sequence of characters, such as “%$#@!98gt”, that starts and ends with double quote characters. This double quote character is represented in a Java program by a double quote character surrounded by two single quote characters (‘“’). When this character appears in a string (in a Java program or a user argument), it is preceded by a backward slash, as in “this is a string with a\" double quote char”. You can assume a string token, like a word or number, is followed by a single space.
3. It will no longer assume that the input is error free. If it finds an unexpected character, it will print the offending character and its position. At this point, it can terminate the program.
4. The scanning class will no longer define any properties. Instead it will implement an iterator interface defined by you.
5. It will do all the scanning at object-initialization time, that is, in (a method called by) a constructor. This means that the tokens must be stored in some list. As in the previous
assignment, store them in an array. You should be able to easily modify your previous class to implement this assignment by making all existing methods non-public and adding some public methods to implement the iterator interface.

Write a main class that uses the above implementation of the iterator interface to print the tokens in each string argument entered by the user. An example run of the main class is shown below. In this and future assignments, use interfaces rather than classes to type variables.

Whether a character is unexpected depends on the kind of extra credit work you have done in previous assignments. For instance, if you require exactly one space between a word and number, then in the string “F1 ”, the character ‘1’ is illegal. On the other hand, if you allow a variable amount of space between tokens, you will separate the string into two tokens. Be sure to show a test case illustrating what you do with this string. In this assignment and others, you will be evaluated on the test cases you show to illustrate various paths through your code. If you do not demonstrate some feature, the TAs are free to assume you have not implemented it.
Assignment 3 Extra Credit
Do not terminate the program on finding the first illegal character. Skip the character and continue scanning from the next token. This is shown in Figure 2.

Assignment 4 Task
Create another implementation of the iterator interface you defined in assignment 3. The only difference between this class and the assignment 3 scanning-class is that it does not store tokens in any list. This means that the scanning task should be done in the implementation of the iterator method that returns the next token. Each time this method is called, the next token should be scanned (and returned). Look at the class example of upper case characters to see how this may be done. As in Assignment 3, write a main class that uses an instance of this class to print the tokens in each string argument.

Assignment 4 Extra Credit
Write a separate method to recognize (and find errors in) each kind of token: word, number, string.

Constraints
You should not use any library code other than what was permitted in previous assignments.
Submission Instructions

1. As in the previous assignment, submit a print out of your code at the start of class on the (early) submission date together with screen shots showing your code working. To print a window, do the following. Click on a window and then press Alt and PrtSc keys together. This will copy the image to a buffer. You can now paste the buffer in a document. To display a portion of an image in Word 2007, select it, go to the Format tab and then crop it. In Word 2003, from what I remember, when you select the image, a picture bar comes up with the crop operation. You are not required to crop images you submit.

2. Upload the assignment directory in blackboard. In general, for all assignments, you should do so by midnight of the day the assignment is due. But do not change the code after you submit it in class.

Good luck!