Java Strings and Lexical Analysis

Consider the process you perform to read. The first thing your brain does is *lexical analysis*, which identifies the distinct words in a sentence.

Consider the job of a compiler (translator)

Source code --> TRANSLATOR --> machine code.

The first thing the translator does is also lexical analysis-- It does the computing equivalent to reading. In this case, it identifies the distinct *tokens* in a program statement or statements. A *lexer* would take input such as:

```
balance4 = balance4 *.05;
```

and return a list of tokens identified by token type:

```
balance4                      identifier
=                              assignment operator
balance4                      identifier
*                              multiplication operator
  .05                          float literal
;                              semicolon
```

Note that the lexer must know things about the language, e.g.,

What is considered a legal identifier (an identifier is a variable/function name).
What is considered a legal float literal?

*Write an English sentence defining what is considered a legal identifier.*

For our sample, the lexer would scan the 'b' of ‘balance4’ then go into a loop to read and collect the rest of the word, stopping when the space was encountered.

*Given how Java defines identifiers, what should the loop to “read and collect the rest of the word” look like? What should the condition be?*

After lexical analysis, the next job of a translator is to *parse* the code-- to see if the tokens form a legal sentence in the programming language. We'll talk about parsing later.
Java Strings

Java Strings are very similar to Python strings:

```java
String s = "";  // note double quotes, not single (').
    s = "abc";
    s = s + "def";  // concatenation
```

Java also has a scalar type called `char`. A char is a single ASCII (Unicode) symbol. The literal value of a char is denoted using single quotes, e.g.

```java
char c = 'a';

if (c == ' ') // is c a space?
```

Underneath the hood, a Java String is a sequence of chars delimited by the end-of-string char ('\0').

*What data members do you think the class String has?*

**In Java, you can’t index a string (as in Python).**

Normal object dot notation is used instead

The String class provides the methods:

```java
    char charAt(int i);
    int length();
```

Example:

```java
    String s = "abcdefg";
    char c = s.charAt(3);  // value of c is 'd'

    int length = s.length(); // value of length is 7
```

**Looping through the characters of a String:**

```java
    int i = 0;
    while (i < s.length())
    {
        char c = s.charAt(i);
        // do something to c.
    }
```

*How would you ask if a char is a letter? A digit?*
Other String Methods (there are many more)
boolean startsWith (String prefix)
String substring(int beginIndex, int endIndex)

Instructor Demo:
http://www.cs.usfca.edu/%7Ewolber/courses/110.f07/javaSamples/WordCounter.java

Write a Java class WordCounter with a main method that

a. reads in a string of text from the user.
b. prints the number of distinct words in the string, where words are delimited by any non-letter. So "big boy" has two distinct words, and "Joe*Bob Jerry" has three.

In-Class Problem
Write a class Lexer with:
• a data member ‘programStatement’ of type String
• a constructor which takes in parameter of type String.
• a method getIdentifer(int index) which returns a string which is the identifier beginning at index.

So if the programStatement is “bob=joe45;”,
getIdentifer(0) would return “bob”.
getIdentifer(4) would return “joe45”

The Lexer Program
GetNextToken method and how it will be used by the parser

Write out an algorithm for the main program and for getNextToken.

What classes should you define for this program?
**class Character**
The Java library has a Character class which provides some static methods that act on the char data type. They can make your job easier.

Its part of java.lang, so remember to import java.lang.*;

Some useful methods for lexical analysis of strings are:

- boolean isDigit(char c)
- boolean isLetter(char c)
- boolean isLetterOrDigit(char c)
- boolean isWhitespace(char c)  // includes tabs and '

For example,

    if (Character.isDigit('3'))

returns true.

**In Class Assignment 2:**

- Rewrite the getIdentifier method of your class Lexer, using Character.isLetterOrDigit.

- Write a getInteger method using isDigit