What You Need In a Scanner

A scanner consists of one function *getNextToken()* that assigns the next token of the program to a variable that is visible to your parser.

You need to create a token type or class. Tokens can be:

Identifiers: These start with a letter and consist of letters, digits, and underscores.

Numbers: These are non-negative integers (so we regard "-23" as two tokens, a minus-token and the number 23. If you want to allow negative numbers as tokens, you may.

Keywords. These are listed on the last page of the BPL reference manual: **int void string if else while return** etc.

Special symbols and punctuaton marks: **; , [ ] { } ( ) < <=** etc.

The **end-of-file token** that indicates the end of the input file has been reached.

A Token object should have instance variables that hold its **string value**, an integer that describes the **kind** of token it is, and an integer for the **line number** of the line of the source file where the token was found.

For the token "kind" I create a bunch of integer constants: T_IF, T_WHILE, T_SEMICOLON, T_LPAREN, and so forth. It doesn't matter what values you assign to these constants as long as the values are unique. Always refer to the token kinds by name rather than value.

For example, if line 26 of our source file is

```
if (num < 23)
```

you should find 6 tokens for this line. The data for these are:

Kind: T_IF	Value: “if”	Line: 26
Kind: T_LPAREN	Value: “(“	Line: 26
Kind: T_ID	Value: “num”	Line: 26
Kind: T_LESS	Value: “<”	Line: 26
Kind: T_NUM	Value: “23”	Line: 26
Kind: T_RPAREN	Value: “)”	Line: 26