

San José State University  
Department of Computer Science

# CS 153

## Concepts of Compiler Design

Fall 2017

Instructor: Ron Mak

### Assignment #2

**Assigned:** Tuesday, September 5

**Due:** Friday, September 15 at 11:59 pm  
Team assignment, 100 points max

#### Java scanner

The purpose of this assignment is to give your programming team experience studying and modifying the frontend code from Chapter 3. Create a new frontend package for the (simplified) Java programming language and plug it into the language-independent framework:

- Create a new package `wci.frontend.java` to contain your new Java language-specific front end classes.
- In this new package, create the necessary Java-specific subclasses that mirror Chapter 3's Pascal-specific subclasses. You can copy and reuse any code you need from the Pascal-specific subclasses.
- Modify the frontend factory class to accommodate your new Java parser and scanner.
- Write a new main class named `Java` to run your Java parser and scanner.
- Make any necessary changes to the language-independent framework classes. But do not break their ability to work with the Pascal-specific subclasses.
- Plus anything else you may discover you need to do.

## Simplified Java tokens

Your Java scanner should recognize the following **reserved word tokens**:

<code>abstract</code>	<code>double</code>	<code>int</code>	<code>long</code>
<code>break</code>	<code>else</code>	<code>long</code>	<code>switch</code>
<code>case</code>	<code>enum</code>	<code>native</code>	<code>super</code>
<code>char</code>	<code>extends</code>	<code>return</code>	<code>this</code>
<code>class</code>	<code>float</code>	<code>short</code>	<code>throw</code>
<code>const</code>	<code>for</code>	<code>package</code>	<code>void</code>
<code>continue</code>	<code>goto</code>	<code>protected</code>	<code>volatile</code>
<code>do</code>	<code>if</code>	<code>static</code>	<code>while</code>

Because Java is a case-sensitive language, `do` and `DO` are not the same.

Here are **special symbol tokens** your Java scanner must recognize:

<code>~</code>	<code>!</code>	<code>@</code>	<code>%</code>	<code>^</code>	<code>&amp;</code>	<code>*</code>	<code>-</code>	<code>+</code>	<code>=</code>
<code> </code>	<code>/</code>	<code>:</code>	<code>;</code>	<code>?</code>	<code>&lt;</code>	<code>&gt;</code>	<code>.</code>	<code>,</code>	
<code>'</code>	<code>"</code>	<code>(</code>	<code>)</code>	<code>[</code>	<code>]</code>	<code>{</code>	<code>}</code>		
<code>++</code>	<code>--</code>	<code>&lt;&lt;</code>	<code>&gt;&gt;</code>	<code>&lt;=</code>	<code>&gt;=</code>	<code>+=</code>	<code>--</code>	<code>*=</code>	<code>/=</code>
<code>==</code>	<code> =</code>	<code>%=</code>	<code>&amp;=</code>	<code>^=</code>	<code>!=</code>	<code>&lt;&lt;=</code>	<code>&gt;&gt;=</code>	<code>  </code>	<code>&amp;&amp;</code>
<code>//</code>	<code>/*</code>	<code>*/</code>							

An **identifier token** consists of one or more letters, digits, and underscores ( `_` ), but the first character must not be a digit. An identifier's length is unlimited.

A **character token** is a single character surrounded by single quotes ( `'` ). Example: `'a'`  
Your Java scanner must recognize that `\` quotes the following character so that it is taken literally. In particular, `'\''` is the single quote character itself. Your scanner must recognize that `'\n'` and `'\t'` represent the line feed and tab characters, respectively.

A **string token** is a sequence of zero or more characters surrounded by double quotes ( `"` ). Example: `"Hello, world."` Your Java scanner must properly handle a character quoted by `\` inside a string literal.

The syntax of integer and floating-point **number tokens** in this simplified Java are the same as Pascal number tokens.

**Comments** can be either:

- Any text surrounded by `/*` and `*/`. Such a comment can span multiple lines.
- Any text following `//` to the end of the current line.

Comments may not be nested.

## Input file

You must run your program with the following input file `javatest.in`. There should be no invisible control characters in the file other than end-of-line characters. You may assume there are no syntax errors in any tokens.

```
/* This is a comment. */
// So is this.

/* Here's a comment
   that spans several
   source lines. */

Two/*comments in*/****a row****/ here.
/* This is /* not a nested comment. */
// Nor is /* this */ one.

{ Not a comment. }

// Word tokens
Hello world
Abstract abstract ABSTRACT aBsTrAcT
What?

// Character tokens
'x' 'A' '\'' 'a' '\n' '\t' '\\\

// String tokens
"Hello, world."
"Hello,\tworld!"
"Hello,\n\"world!\""
"It's Friday!"
"" "\

// Special symbol tokens
+ - * / := . , ; : = <> < <= >= > ( ) [ ] { } } ^ ..
<<= >>=
:=<>=<=>>====
```

## Output

Your output should be similar in format to the output of the Pascal version in Listing 3-4 of the book.

## Running your program

Add a new main class `Java`. Similarly to how you run the book's Program 3: Pascal Tokenizer, you should be able to run your Java Tokenizer with a command line like:

```
java -classpath classes Java compile javatest.in
```

Leave in the original Pascal main program and scanner. After you add the Java-specific subclasses, you should still be able to process a Pascal program with the same code base:

```
java -classpath classes Pascal compile hello.pas
```

In other words, you're building a frontend that can process two programming languages. (In a real-world scenario, you would write the front end so that at run time, it dynamically loads the code that is specific to the source language you're compiling.)

## What to submit

This is a team assignment. Each team turns in one assignment and each team member will get the same score.

- A zip file containing all your Java source files. Name the zip file after your team name. Example: `SuperCoders.zip`
- A text file containing the output from running your program with the `javatest.in` input file. You can include this file in your zip file.
- A paragraph or two describing:
  - Any assumptions you made.
  - Anything clever you may have done in your program.You can include this report in your zip file.

Submit into Canvas: **Assignment #2. Java Scanner**

## Rubric

Your program will be graded according to these criteria:

Criteria	Maximum points
<b>Token recognition</b> <ul style="list-style-type: none"><li>• identifiers</li><li>• reserved words</li><li>• numbers</li><li>• special symbols</li><li>• characters</li><li>• strings</li><li>• comments</li></ul>	<b>70</b> <ul style="list-style-type: none"><li>• 10</li><li>• 10</li><li>• 10</li><li>• 10</li><li>• 10</li><li>• 10</li><li>• 10</li></ul>
<b>Good program design</b> <ul style="list-style-type: none"><li>• New front end for Java</li><li>• Preserved old front end for Pascal</li></ul>	<b>30</b> <ul style="list-style-type: none"><li>• 15</li><li>• 15</li></ul>