CS 316 (Kong): TinyJ Assignment 1

To be submitted *no later than*: Tuesday, December 1. [**Note**: I expect euclid to be up until midnight that evening, but there is no guarantee that it will be: If euclid unexpectedly goes down after 6 p.m., the deadline will ***not*** be extended. If you try to submit after 6 p.m. that evening and find that euclid is down, you may have to make a ***late*** submission!] This assignment counts 1.5% towards your grade if the grade is computed using Rule A.

The TinyJ language is an extremely small subset of Java. Every valid TinyJ program is a valid Java program, and has the same semantics whether it is regarded as a TinyJ or a Java program. The syntax of TinyJ is given by the following EBNF rules: A Java program is a TinyJ program if and only if it can be generated by these EBNF rules, except that TinyJ doesn't support method name overloading, program arguments, "**return;**" statements within the **main**() method, and ints that are $\geq 2^{31}-2^{16}$ = 2,147,418,112. Reserved words of TinyJ are shown in boldface. Some names used by Java library packages, classes, and their methods (e.g., **java**, **Scanner**, and **nextInt**) are reserved words of TinyJ, as is **main**. Otherwise, IDENTIFIER here means any Java identifier consisting of ASCII characters.

```
<program>             ::=  [<importStmt>] class IDENTIFIER '{' {<dataFieldDecl>}
                               <mainDecl> {<methodDecl>} '}'
<importStmt>          ::=  import java . util . Scanner ;
<dataFieldDecl>       ::=  static <varDecl>
<varDecl>             ::=  int <singleVarDecl> { , <singleVarDecl>} ;
                        |   Scanner IDENTIFIER = new Scanner '(' System . in ')' ;
<singleVarDecl>       ::=  IDENTIFIER { '[' ']' } [ = <expr3> ]
<mainDecl>            ::=  public static void main '(' String IDENTIFIER '[' ']' ')'
                             <compoundStmt>
<methodDecl>          ::=  static ( void | int {'[' ']'}) IDENTIFIER
                             '(' <parameterDeclList> ')' <compoundStmt>
<parameterDeclList>   ::=  [<parameterDecl> { , <parameterDecl> }]
<parameterDecl>       ::=  int IDENTIFIER {'[' ']'}
<compoundStmt>        ::=  '{' { <statement> } '}'
<statement>           ::=  ; | return [<expr3>] ; | <varDecl> | <assignmentOrInvoc>
                        |  <compoundStmt> | <ifStmt> | <whileStmt> | <outputStmt>
<assignmentOrInvoc>   ::=  IDENTIFIER ( { '['<expr3>']' } = <expr3> ; | <argumentList> ; )
<argumentList>        ::=  '('[<expr3>{,<expr3>}]')'
<ifStmt>              ::=  if '(' <expr7> ')' <statement> [else <statement>]
<whileStmt>           ::=  while '(' <expr7> ')' <statement>
<outputStmt>          ::=  System . out . (  print '(' <printArgument> ')' ;
                                            | println '(' [<printArgument>] ')' ;
                                           )
<printArgument>       ::=  CHARSTRING | <expr3>
<expr7>               ::=  <expr6> { '|' <expr6> }
<expr6>               ::=  <expr5> { & <expr5> }
<expr5>               ::=  <expr4> {(== | !=) <expr4> }
<expr4>               ::=  <expr3> [(> | < | >= | <=) <expr3> ]
<expr3>               ::=  <expr2> {(+ | -) <expr2> }
<expr2>               ::=  <expr1> {(* | / | %) <expr1> }
<expr1>               ::=  '(' <expr7> ')' | (+|-|!) <expr1> | UNSIGNEDINT | null
                        |   new int '[' <expr3> ']' { '[' ']' }
                        |   IDENTIFIER ( . nextInt '(' ')' | [<argumentList>]{'[' <expr3> ']'} )
```

This is the first of three TinyJ assignments. After completing all three assignments you will have a program that can compile any TinyJ program into a simple virtual machine code, and then execute the virtual machine code it has generated. (Execution should produce the same run-time behavior as you would get if you compiled the same TinyJ program using `javac` into a `.class` file and then executed that `.class` file using a Java VM.) **<u>There will be exam questions relating to the TinyJ assignments, which may count 25 – 50% towards your grade</u>**.

TinyJ Assignment 1 will not deal with compilation of TinyJ programs, nor with execution of virtual machine code, but only with *syntax analysis* of TinyJ programs. The goal of TinyJ Assignment 1 is to complete a program that will:

(a) determine whether its input file is a `<program>` according to the above EBNF rules, and

(b) output a parse tree of the input file, if the input file is a `<program>`.

Regarding (a), note that a `<program>` is a ***syntactically*** valid TinyJ program, but may contain errors like "undeclared variable" or "array index out of range". A "sideways" representation of ordered trees, described below, will be used for (b).

A Sideways Representation of an Ordered Rooted Tree *T*

If *T* has just one node, then representation of *T* = the unique node of *T*

Otherwise, representation of *T* = the root of *T*

representation of the 1st subtree of the root of *T*
representation of the 2nd subtree of the root of *T*
...
representation of the last subtree of the root of *T*
`... node has no more children`

In this sideways representation, sibling nodes always have the *same* indentation, but each non-root node is further indented than its parent; *the indentation of a node is proportional to the depth of that node in the tree.* Here are the "ordinary" and the "sideways" representations of a tree:

```
              <expr4>                          <expr4>
                 |                              <expr3>
              <expr3>                            <expr2>
            /    |    \                           <expr1>
     <expr2>     +    <expr2>                      UNSIGNEDINT
        /              /   |   \                    ... node has no more children
    <expr1>      <expr1> * <expr1>                ... node has no more children
       |             |         |                 +
   UNSIGNEDINT IDENTIFIER UNSIGNEDINT            <expr2>
                                                  <expr1>
                                                   IDENTIFIER
                                                   ... node has no more children
                                                  *
                                                  <expr1>
                                                   UNSIGNEDINT
                                                   ... node has no more children
                                                  ... node has no more children
                                                 ... node has no more children
                                                ... node has no more children
```

How to Install the TinyJ Assignment 1 Files on euclid, venus, and Your PC

Do 1 – 5, and optionally 6 – 11, before our class on **Monday, November 16**. (See **Seven Files You May Need to Refer to** on p. 3.) Remember that Unix/Linux file and command names are ***case-sensitive*** when following the instructions below!

1. Login to ***euclid*** and enter: **`/users/kong300/316/TJ1setup`** [The `1` in `TJ1setup` is the ***digit*** `1`, not the letter `l`.]

2. Wait for the line "`TJ1setup done`" to appear on the screen, and then enter the following command on ***euclid***:

   ```
   java -cp TJ1solclasses:.  TJ1asn.TJ  CS316ex12.java  12.sol
   ```

   Note the ***period*** after the ***colon*** in this command. This command executes my solution to this assignment with `CS316ex12.java` as the input file and `12.sol` as the output file. A listing of `CS316ex12.java` should be displayed on the screen, and `12.sol` should contain a sideways representation of the program's parse tree afterwards. **There should not be any error message**. To view the tree, you can use **`less 12.sol`** or just open `12.sol` in an editor.

3. Logout from ***euclid*** and login to ***venus***.

4. Enter the following on ***venus***: **`/home/faculty/ykong/TJ1setup`**
   [Again, the `1` in `TJ1setup` is the ***digit*** `1`, not the letter `l`.]

5. **Repeat step 2 above on *venus*.**

The following 6 steps are needed ***only if*** you are interested in doing TinyJ assignments on your PC rather than euclid or venus. (Regardless of where you do your work, you must submit on ***euclid***.) These instructions assume you are using Oracle's JDK.

6. In a **cmd.exe** (command prompt) window* on your PC, enter the following: **`md c:\316java`**

   *You can open a **cmd.exe** window on your Windows PC as follows:
   1. Type **Win**-`r` (i.e., hold down the Windows key and type `r`) to open the Run dialog box. 2. Type `cmd` into the Open: textbox and press ENTER.

7. Enter **`javac -version`** in the **cmd.exe** window. If you get an error message or if the version number that is printed is older than **1.6**, then download and install a newer version of the Java JDK (e.g., the Java SE 8 JDK) from the URL
   `http://www.oracle.com/technetwork/java/javase/downloads/index.html` and set the PATH variable as explained at[†]
   `http://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_jdk_install.html#BABGDJFH`

   [†]If you have difficulty with **step 1** of these instructions for setting the PATH, try the following instead of that step:
   1. Type **Win**-`r` to open the Run dialog box. 2. Type `control sysdm.cpl` into the Open: textbox and press ENTER.

8. Make **`c:\316java`** your working directory by entering the following in the **cmd.exe** window: **`cd /d c:\316java`**

9. Using an scp or sftp client, download **`TJ1asn.jar`** from your home directory on ***venus*** or ***euclid*** into the **`c:\316java`** folder on your **PC**. *For example*, if the PuTTY programs have been installed in the **`c:\Program Files (x86)\PuTTY`** folder on a PC running a 64-bit version of Windows, or in the **`c:\Program Files\PuTTY`** folder on a PC running a 32-bit version of Windows (which can be done, e.g., by downloading the file **`putty-0.66-installer.exe`** from **`http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`** and running that installer), and assuming **`c:\316java`** is your working directory in the **cmd.exe** window (see step 8), when you are connected to the Internet you can download **`TJ1asn.jar`** by entering the appropriate one of the following in the **cmd.exe** window:
   ```
   "c:\program files (x86)\putty\pscp"  xxxxx316@euclid.cs.qc.cuny.edu:TJ1asn.jar  .
   "c:\program files\putty\pscp"  xxxxx316@euclid.cs.qc.cuny.edu:TJ1asn.jar  .
   ```
   Here **`xxxxx316`** means your **`euclid`** username. *Note the __double quotes__ in* **`"c:\program ... pscp"`** *and the space followed by a __period__ at the end of the line!* [***<u>Alternatively</u>***, you can logon to ***euclid*** and send **`TJ1asn.jar`** to yourself as an *e-mail attachment* by entering the following on euclid: **`pine -attach TJ1asn.jar`** *your-email-address*
   After pine starts up, enter CTRL`x y` to send the file. But if you use this method you will need to save the attached file (i.e., the file **`TJ1asn.jar`**) into the **`C:\316java`** directory on your PC.]

10. Enter the following ***two*** commands in the **cmd.exe** window:
    ```
    jar xvf TJ1asn.jar
    javac -cp . TJ1asn\TJ.java
    ```

11. Enter the following command in the **cmd.exe** window:
    ```
    java -cp TJ1solclasses;. TJ1asn.TJ  CS316ex12.java  12.sol
    ```
    <u>The comments on step 2 also apply here</u>, except that a ***semicolon*** rather than a colon precedes the period. A version of `less` for Windows PCs is available at: **`http://gnuwin32.sourceforge.net/packages/less.htm`**

Seven Files You May Need to Refer to in Class Starting on Monday, November 16

<u>Either</u> bring printouts of these seven files to class, ***<u>or</u>*** bring a laptop / tablet to class and be ready to *quickly* bring any of these seven files up on your screen:

From your `TJ1asn` directory on euclid:
```
OutputFileHandler.java.txt    Parser.java.txt    SourceFileErrorException.java.txt    TJ.java.txt
```
From your `TJlexer` directory on euclid (the `l` in `TJlexer` is the ***<u>letter</u>*** `l`, not the digit 1):
```
LexicalAnalyzer.java.txt      SourceHandler.java.txt      Symbols.java.txt
```
These are the source files of the program, with **<u>line numbers added</u>**. The actual source files (without line numbers) are in the same directories and have the same names, but their extension is `.java`. ***If you have done steps 6 – 11 above, you can find the same files in* `C:\316java\TJ1asn` *and* `C:\316java\TJlexer` *on your PC, and these files can be opened using, e.g., any of the editors recommended in the last paragraph of p. 4 of the Lisp Assignment 2 document. Otherwise, you can e-mail the files to yourself*—e.g., *you can send* `TJ1asn/TJ.java.txt` *to yourself by entering the following on euclid*:**
`pine -attach TJ1asn/TJ.java.txt` *your-email-address* **[*After pine starts up, enter* CTRL`x y` *to send the file.*]**

How to Execute My Solution to This Assignment

Steps 1 and 4 put 16 files named `CS316ex`*k*`.java` (*k* = 0 – 15) into your home directories on ***euclid*** and ***venus***. These are all valid TinyJ source files. If you did step 10, it will have put copies of the same 16 files on your PC. You should be able to execute *my* solution to this assignment either on ***euclid*** or on ***venus*** by entering the following command:
```
java -cp  TJ1solclasses:. TJ1asn.TJ  TinyJ-source-file-name   output-file-name
```
[Your current working directory has to be your home directory for this to work.] This should also work in a cmd.exe window on your PC if you have done steps 6 – 11, **<u>except that you need a *semicolon* instead of a colon after</u>** `TJ1solclasses` on a PC:
```
java -cp  TJ1solclasses;. TJ1asn.TJ  TinyJ-source-file-name   output-file-name
```
[Your working directory has to be `C:\316java` for this to work (see step 8).]

How to Do TinyJ Assignment 1

The file `TJ1asn/Parser.java` is incomplete. It was produced by taking a complete version of that file and replacing parts of the code with comments of the follwing two forms:

```
        /* ???????? */            or (in two places)     /* ????????
                                                                  default: throw ...
                                                         */
```

To complete this assignment, replace every such comment in `TJ1asn/Parser.java` with appropriate code, and recompile the file. On venus or euclid, you can use any text editor to edit the file. If you are working on your PC, do ***not*** use Notepad as your editor; I suggest you ***use one of the editors listed in the last paragraph on p.* 4 *of the* Lisp Assignment 2 document**. (For the second type of comment, the appropriate code should include the `default: throw ...` statement.)

*Do **not** put `Parser.java` or `Parser.class` into any directory other than `TJ1asn`. Do **not** change or move other `.java` and `.class` files.*

To recompile `TJ1asn/Parser.java` after editing it, enter the following command:

```
javac  -cp  .  TJ1asn/Parser.java
```

IMPORTANT: If you are doing this on ***venus*** or ***euclid***, your current working directory has to be your home directory. If you are doing this on your PC (in a **cmd.exe** window), your working directory has to be **`c:\316java`** (see installation step 8); otherwise `javac` will not be able to find other classes that are used in `Parser.java`!

As stated on p. 3 of the first-day handout, ***keep backups*** of your edited versions of `Parser.java` on **venus** and elsewhere.

How to Test Your Solution

Test your completed version of `Parser.java` by recompiling it, and then executing the **`TJ1asn.TJ`** program with each of the 16 files `CS316ex`*k*`.java` (*k* = 0 – 15) as the TinyJ source file and *k*`.out` as the output file. You can do this by entering the following two commands:

```
javac  -cp  .  TJ1asn/Parser.java
java  -cp .   TJ1asn.TJ  CS316exk.java  k.out
```

If you are doing this on ***venus*** or ***euclid***, your current working directory has to be your home directory. If you are doing this on your PC (in a **cmd.exe** window), your working directory has to be **`c:\316java`** (see installation step 8).

If your program is correct then in each case the output file *k*`.out` should be identical to the output file *k*`.sol` that is produced by running my solution with the same source file as follows:

```
java -cp  TJ1solclasses:. TJ1asn.TJ  CS316exk.java  k.sol      [on euclid or venus]
java -cp  TJ1solclasses;. TJ1asn.TJ  CS316exk.java  k.sol      [on a PC]
```

On ***euclid*** and ***venus***, use **`diff -c`** to compare the output files produced by your and my solutions. (This outputs a report of the differences, if any, between the two files.) On a PC, use **`fc /n`** to compare files. For example, the commands **`diff -c  k.sol k.out > k.dif`** [on **venus** or **euclid**] and **`fc /n  k.sol k.out > k.dif`** [on a PC] output to *k*`.dif` the differences between *k*`.sol` and *k*`.out`. (If your solution is correct, ***there should be no differences***.)

How to Submit a Solution to This Assignment

This assignment is to be submitted *no later than* **Tuesday, December 1**. It counts 1.5% towards your grade if the grade is computed using Rule A. [**Note**: If **euclid** unexpectedly goes down after 6 p.m. on the due date, the deadline will ***not*** be extended. Try to submit no later than noon that day, and on an earlier day if possible.] To submit:

1. Add a comment at the beginning of your completed version of `Parser.java` that gives your name and the names of the students you worked with (if any). As usual, you may work with up to two other students.
2. Leave your final version of `Parser.java` on ***euclid*** in your `TJ1asn` directory, so it replaces the original version of `Parser.java`, before midnight on the due date. When two or three students work together, ***each*** of the students must leave the completed file in his/her directory. If you are working on venus or your PC, you can transfer `Parser.java` to your `TJ1asn` directory on euclid **by following the instructions on the next page**.
3. Be sure to test your submission on euclid. Note that if your modified version of `Parser.java` cannot even be compiled without error on ***euclid***, then you will receive no credit at all for your submission!

IMPORTANT: Do NOT open your submitted file `Parser.java` in an editor on euclid after the due date, unless you are resubmitting a corrected version of your solution as a *late* submission.

How to Transfer `TJ1asn/Parser.java` from `venus` or a PC to `euclid`’s `TJ1asn` Directory

The following instructions assume that **`xxxxx316`** is your username on **`euclid`**.

If you are working on **`venus`**, and your current working directory is your home directory, enter the following command to transfer **`TJ1asn/Parser.java`** to your **`TJ1asn`** directory on **`euclid`**:

```
      scp  TJ1asn/Parser.java  xxxxx316@euclid.cs.qc.edu:TJ1asn
```

You will be asked to enter your euclid password.

If you are working on a PC that is running a 64-bit version of Windows and the PuTTY programs have been installed into the **`c:\Program Files (x86)\PuTTY`** folder, or you are working on a PC that is running a 32-bit version of Windows and the PuTTY programs* have been installed into the **`c:\Program Files\PuTTY`** folder, then you can transfer **`TJ1asn/Parser.java`** into your **`TJ1asn`** directory on **`euclid`** by entering the appropriate one of the following commands in a **cmd.exe** window:

64-bit Windows:

```
   "c:\program files (x86)\putty\pscp"  TJ1asn/Parser.java  xxxxx316@euclid.cs.qc.edu:TJ1asn
```

32-bit Windows:

```
   "c:\program files\putty\pscp"  TJ1asn/Parser.java   xxxxx316@euclid.cs.qc.edu:TJ1asn
```

The double quotes and the backslashes in **`"c:\program ... pscp"`** are needed! You will be asked to enter your euclid password.

*You can get the PuTTY programs by downloading the installer file **putty-0.66-installer.exe** from **http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html** and then running this installer on your PC.