CSCI 316 (Kong): TinyJ Assignment 1

To be submitted *no later than*: Friday, May 3. [**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! Try to submit no later than noon that day, and ***on an earlier day if possible***. TinyJ Assignment 2 will be provided to you before **Wednesday, April 17**.]
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 EBNF specification that is shown below. *In this EBNF specification each terminal is a token of TinyJ, and each nonterminal <X> denotes the set of all sequences of tokens that are **syntactically** valid for the TinyJ construct X.* In particular, a piece of source code is a *syntactically* valid TinyJ program if and only if its sequence of tokens belongs to the language generated by this EBNF specification. A piece of source code is a valid TinyJ program if and only if it is *both* a syntactically valid TinyJ program *and* a valid Java 8 program, with a few exceptions: TinyJ does ***not*** allow non-decimal (i.e., hexadecimal, octal, or binary) or long integer literals, underscores in integer literals, method name overloading, program arguments, printing of Boolean values, "**return;**" statements within the **main**() method, escape sequences other than \n, \\, and \", and ints that are $\geq 2^{31}-2^{16} = 2{,}147{,}418{,}112$.

Reserved words of TinyJ are shown in boldface in this EBNF specification. 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.) **There will be exam questions relating to the TinyJ assignments.**

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 if the sequence of tokens of its input file belongs to <program> (as defined by the above EBNF rules), and

(b) output a parse tree of the sequence of tokens of its input file, if that sequence belongs to <program> .

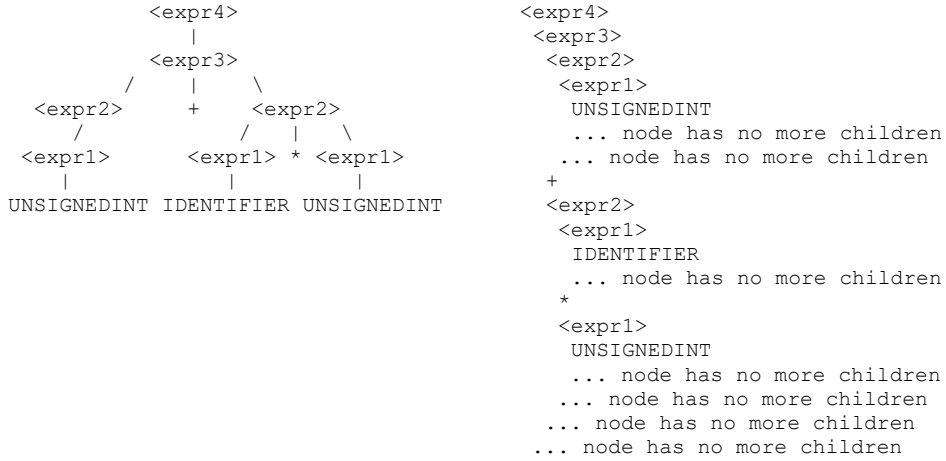
Regarding (a), note that the sequence of tokens of the input file belongs to <program> if, and only if, the input file is a *syntactically* valid TinyJ program. However, a syntactically valid TinyJ program may still contain errors like "undeclared variable" or "array index out of range". A "sideways" 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:



How to Install the TinyJ Assignment 1 Files on euclid, mars, and (optionally) Your PC / Mac

Do 1 – 5, and optionally 6 – 11, before our class on **Monday, April 15** (and preferably before our class on Wednesday, April 10). 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 parse tree, you can use **`less 12.sol`** or just open `12.sol` in an editor.

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

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

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

The following 6 steps are needed ***only if*** you are interested in the possibility of doing TinyJ assignments on your PC or Mac rather than **euclid** or **mars**; step 9 assumes your PC / Mac is connected to the qwifi-secured wireless network or connected to the Queens College VPN. (**Important**: Regardless of where you do the assignments, you must submit all the assignments on ***euclid*** and test your submissions on ***euclid***.) While many students in previous semesters were able to do the TinyJ assignments on a PC or Mac, I do not guarantee that you will be able to do so: *All students who try to do the assignments on a PC or Mac must be prepared to switch to working on mars or euclid if they run into difficulties.*

6. Open a powershell / terminal window on your PC / Mac and enter the following at its prompt: **`javac -version`**
   If you get an error message after entering **`javac -version`**, or if the version number that is printed is older than **1.8.0**, install a new version of the Java JDK—e.g., JDK 21 from https://www.oracle.com/technetwork/java/javase/downloads/index.html. (After installing the JDK on a PC, update the PC's System PATH environment variable so its first directory is the directory that contains the JDK's `jar.exe` application; for a typical installation of JDK 21, `c:\program files\java\jdk-21\bin` is the directory that should be added to your PC's System PATH. See, e.g., https://www.computerhope.com/issues/ch000549.htm if you don't know how to edit your PC's System PATH.)
7. In the powershell / terminal window, enter the following: **`mkdir ~/316java`**
8. Make **`~/316java`** your working directory by entering the following in the powershell / terminal window: **`cd ~/316java`**
9. Use an scp or sftp client to copy **`TJ1asn.jar`** from your home directory on ***mars*** or ***euclid*** into the **`~/316java`** folder. If **`~/316java`** is your working directory in the powershell / terminal window (see step 8), then you can do this by entering the following command in that window: **`scp xxxxx_yyyy316@euclid.cs.qc.cuny.edu:TJ1asn.jar .`**
   Here **`xxxxx_yyyy316`** means your **`euclid`** username. *Note the space followed by a **period** at the end of this command!*
10. Enter the following ***two*** commands in the powershell / terminal window:
```
jar xvf TJ1asn.jar
javac -cp . TJ1asn/TJ.java
```
11. Enter the appropriate one of the following commands in the powershell / terminal window:
    On a PC: **`java -cp "TJ1solclasses;." TJ1asn.TJ CS316ex12.java 12.sol`**
    On a Mac: **`java -cp TJ1solclasses:. TJ1asn.TJ CS316ex12.java 12.sol`**
    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 parse tree, you can enter the command **`more 12.sol`** on a PC or **`less 12.sol`** on a Mac.

Important Files That will be Available to You After You Have Done Steps 1 – 5 Above

From your `TJ1asn` directory on **euclid** and **mars**:
`OutputFileHandler.java.txt` `Parser.java.txt` `SourceFileErrorException.java.txt` `TJ.java.txt`
From your `TJlexer` directory on euclid and mars (the `l` in `TJlexer` is the ***letter*** `l`, not the digit 1):
`LexicalAnalyzer.java.txt` `SourceHandler.java.txt` `Symbols.java.txt`

These are the source files of the program, with **line numbers added**. (The actual source files (without line numbers) are in the same directories and have the same names, but their extension is `.java`.) The files can be viewed on **euclid** or **mars** using the **`less`** file viewer—e.g., enter the command **`less TJ1asn/Parser.java.txt`** to view `Parser.java.txt`, and enter the command **`less TJlexer/Symbols.java.txt`** to view `Symbols.java.txt`.

If you have done steps 6 – 11 above, the same files will be in `~/316java/TJ1asn` and `~/316java/TJlexer` on your PC or Mac; they can be viewed using **`less`** on a Mac (e.g., enter **`less ~/316java/TJ1asn/Parser.java.txt`** in a terminal window on a Mac to view `Parser.java.txt`) and can be viewed using, e.g., Notepad++ or VS Code on a PC.

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 ***mars***. 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 or Mac.

You should be able to execute *my* solution to this assignment either on ***euclid*** or on ***mars*** 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.]

If you have done steps 6 – 11 on a Mac, then the above command should also work in a terminal window on your Mac if your working directory is **`~/316java`** (see step 8). If you have done steps 6 – 11 on a PC, then the following similar command (which has **`;.`** instead of **`:.`**) should work in a powershell window on your PC if **`~/316java`** is your working directory:
`java -cp "TJ1solclasses;." TJ1asn.TJ` ***`TinyJ-source-file-name output-file-name`***

See steps 2 and 11 above for concrete examples of these commands!

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 following 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 **mars** or **euclid**, you can use the **`nano`**, **`vim`**, or **`emacs`** editor to edit the file; **`nano`** or **`vim`** could also be used on a Mac in a terminal window. If you are working on your PC, do ***not*** use Notepad as your editor; you can use Notepad++ or VS Code. (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 ***mars*** or ***euclid***, your current working directory has to be your home directory. If you are doing this on your PC or Mac (in a powershell / terminal window), your working directory has to be **`~/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 announcements, ***keep a backup copy*** of your edited version of `Parser.java` on **mars** and another backup copy on a different machine.

How to Test Your Solution

To test your completed version of `Parser.java`, first recompile it using **`javac -cp . TJ1asn/Parser.java`** and then execute **`TJ1asn.TJ`** with each of the 16 files `CS316ex`*k*`.java` (*k* = 0 – 15) as the TinyJ source file and *k*`.out` as the output file, as follows: **`java -cp . TJ1asn.TJ CS316ex`***k*`.java` ***k*`.out`**

If you are doing this on ***mars*** or ***euclid***, your current working directory has to be your home directory. If you are doing this on your PC or Mac (in a powershell / terminal window), your working directory has to be **`~/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 CS316ex`*k*`.java` ***k*`.sol`** [on **euclid**, **mars**, or a Mac]

`java -cp "TJ1solclasses;." TJ1asn.TJ CS316ex`*k*`.java` ***k*`.sol`** [on a PC]

On ***euclid***, ***mars***, or a Mac, you can 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, you can use **`fc.exe /n`** instead. For example, the commands **`diff -c `*k*`.sol `*k*`.out > `*k*`.dif`** [on **mars**, **euclid**, or Mac] and **`fc.exe /n `*k*`.sol `*k*`.out > `*k*`.dif`** [on a PC] output to *k*`.dif` the differences between *k*`.sol` and *k*`.out`. (You can view *k*`.dif` using the command **`less `*k*`.dif`** on **euclid**, **mars**, or a Mac, or using the command **`more `*k*`.dif`** on a PC. If your solution is correct, then the file ***k*`.dif`** should contain nothing if it was produced by **`diff -c`** or contain "FC: no differences encountered" if it was produced by **`fc.exe /n`**.)

How to Submit a Solution to This Assignment

This assignment is to be submitted *no later than* the due date stated on p. 1. [**Note**: If **euclid** unexpectedly goes down after 6 p.m. on this 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, but see the remarks about this on p. 3 of the first-day announcements document.
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 his/her completed file in his/her directory. If you are working on **mars** or your PC / Mac, you can copy the file `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***—see the **How to Test Your Solution** instructions above. 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. Also do not execute `mv`, `chmod`, or `touch` with your submitted file as an argument after the due date. (However, it is OK to view a submitted file using the `less` file viewer after the due date.) Remember that, as stated on page 3 of the first-day announcements document, you are required to keep a *backup copy* of your submitted file on mars—see the final paragraph on the next page.

How to Copy `TJ1asn/Parser.java` from `mars` or a PC / Mac to `euclid`'s `TJ1asn` Directory

The instructions below will *NOT* work if you haven't yet done installation steps 1 and 2 above!

The instructions assume that **`xxxxx_yyyy316`** is your **euclid** username.

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

```
scp  TJ1asn/Parser.java  xxxxx_yyyy316@euclid.cs.qc.cuny.edu:TJ1asn
```

You will be asked to enter your euclid password.

If you are working on a PC or Mac that is connected to the qwifi-secured wireless network or connected to the Queens College VPN, then this same **`scp`** command can be used in a powershell / terminal window to copy the file **`TJ1asn/Parser.java`** from your PC or Mac into your **`TJ1asn`** directory on **`euclid`**, provided that your working directory is **`~/316java`** (see installation step 8). Again, you will be asked to enter your euclid password.

IMPORTANT REMINDER: After you have copied **`TJ1asn/Parser.java`** to your **`TJ1asn`** directory on **`euclid`**, be sure to ***test your code on euclid***—see the **How to Test Your Solution** instructions on the previous page. (It is ***not*** enough to have tested your code on mars or your PC / Mac, because testing on a machine other than euclid does not test the file you actually submitted!)

As stated on page 3 of the 1st-day announcements document, you are required to keep a backup copy of your submitted file on **mars**. You can enter the following command on **euclid** to put a copy of the file on **mars**:

```
scp  TJ1asn/Parser.java  your mars username@mars.cs.qc.cuny.edu:
```

The colon at the end of this command is needed!