C SCI 316 (Kong): TinyJ Assignment 2

Your assignment is to complete a compiler which does all of the following whenever its input is a syntactically valid TinyJ source file:

1. It checks that declarations and uses of identifiers in the source file are consistent with Java's scope rules.
2. As long as no errors are detected in the source file, it translates the source file into a sequence of instructions for a stack-based virtual machine whose instruction set is given below. At the same time, it writes to the output file an "enhanced parse tree" of the source file; this shows the static address or the stackframe offset that the compiler has allocated to each int or array reference variable, the start address of the code generated for each method, and the time at which each instruction is generated.
3. If no errors are detected in the source file then a list of the instructions generated is also written to the output file.

This assignment is to be submitted *no later than* **Thursday, December 8**, ***but you should aim to finish the assignment sooner to leave yourself more time to work on TinyJ assignment 3*** (which you will receive ***before*** our class on **Monday, Dec. 5**) and to prepare for Exam 2 and the Final Exam. [**Note**: If euclid goes down after 6 pm on the due date, the deadline will ***not*** be extended. Try to submit no later than noon on that day, and sooner if you can.]

The 35 virtual machine instructions that may be generated by the compiler are shown below. A specification of the effects of executing these instructions is given on pp. 5 – 6 of the following document:
https://phantom.cs.qc.cuny.edu/kong/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf

Operation	**Operand 1**	**Operand 2**
`STOP`		
`PUSHNUM`	`<integer>`	
`PUSHSTATADDR`	`<address of static variable>`	
`PUSHLOCADDR`	`<local variable or parameter's stackframe offset>`	
`LOADFROMADDR`		
`SAVETOADDR`		
`WRITELNOP`		
`WRITEINT`		
`WRITESTRING`	`<address of first char>`	`<address of last char>`
`READINT`		
`CHANGESIGN`		
`NOT`		
`ADD`		
`SUB`		
`MUL`		
`DIV`		
`MOD`		
`AND`		
`OR`		
`EQ`		
`LT`		
`GT`		
`NE`		
`GE`		
`LE`		
`JUMP`	`<address of target instruction>`	
`JUMPONFALSE`	`<address of target instruction>`	
`CALLSTATMETHOD`	`<address of method's first instruction>`	
`INITSTKFRM`	`<no. of locations needed for local vars declared in the method's body>`	
`RETURN`	`<no. of parameters the method has>`	
`HEAPALLOC`		
`ADDTOPTR`		
`PASSPARAM`		
`DISCARDVALUE`		
`NOP`		

How to Install the Already-Written Parts of the Program (Assuming You Have *Already* Followed the Installation Instructions Provided with TinyJ Assignment 1)

1. Login to **euclid** and enter the following command: **`/users/kong300/316/TJ2setup`**
   Wait for **`TJ2setup done`** to appear on the screen. **Important**: There should be no error messages!
2. ***Logout from euclid***. Then ***login to venus*** and enter: **`/home/faculty/ykong/TJ2setup`**
   Wait for **`TJ2setup done`** to appear on the screen. Again, there should be no error messages!

Also do the next 3 steps **if** you did assignment 1 on your PC and plan to do this assignment on your PC too:

3. In a cmd.exe (command prompt) window on the PC, make **`c:\316java`** your working directory by entering the following command: **`cd /d c:\316java`**
4. Either use an scp or sftp client to download the file **`TJasn.jar`** from your home directory on *euclid* or *venus* to the **`c:\316java`** folder on your PC, or e-mail that file to yourself and save it in your **`c:\316java`** folder. [See installation step 9 on p. 3 of the Assignment 1 document, but substitute the filename **`TJasn.jar`** for **`TJ1asn.jar`** when you follow those instructions.]
5. On your PC, enter the following three commands in the cmd.exe window (with **`c:\316java`** as your working directory):

```
jar   xvf   TJasn.jar
javac  -cp .  TJasn/TJ.java
javac  -cp .  TJasn/virtualMachine/*.java
```

Some or all of the following files will be considered in class:

`ParserAndTranslator.java.txt` (in the TJasn directory on venus, euclid, or your PC).
The 8 other `.java.txt` files in the TJasn directory and its virtualMachine subdirectories.
(There are 3 `.java.txt` files in TJasn and 6 in virtualMachine.)

How to Do This Assignment

The only file you need to change is `TJasn/ParserAndTranslator.java`. In this file, each /* ???????? */ comment must be replaced with appropriate code. *For most of these comments (specifically, the ones on lines 511 − 723), a good way to begin is to **first copy over** code you wrote for Parser.java in the previous assignment. (Do **not** open the Parser.java file you submitted; copy the code from **one of your two backup copies** of that file!) **Then** make changes, so appropriate virtual machine instructions are generated.*

Note: For the comment on line 511, you should only have to copy 3 statements! Do ***not*** copy over the lines of TJ1asn/Parser.java that correspond to lines 482–3, 500, 502–4, 507, and 513 in TJasn/ParserAndTranslator.java, and also do ***not*** copy the lines corresponding to lines 515, 519, and 537–8 in that file.

To recompile `TJasn/ParserAndTranslator.java` after you have edited it, enter:

```
javac -cp .  TJasn/ParserAndTranslator.java
```

On euclid and venus, this command assumes your working directory is your home directory. On your PC (in a cmd.exe window) this command assumes your working directory is **`c:\316java`**.

To have a good chance of finishing before the submission deadline, I recommend you do the following:

- For the /* ???????? */ comments on lines 511 − 723, copy over code from the corresponding parts of TinyJ Assignment 1 as soon as you have finished that assignment.
- Fill in the /* ???????? */ gaps that were on lines 638 – 682 ***before*** **Friday, December 2**.

This should leave you enough time to fill in the other /* ???????? */ gaps—i.e., the gaps that were on lines 492, 495, 511, 549, 593, 610, 627, and 723—well before the submission deadline. Some **hints** are given on the last few pages of:
https://phantom.cs.qc.cuny.edu/kong/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf

How to Test Your Solution

First recompile your program as follows: `javac -cp .  TJasn/ParserAndTranslator.java`
Then test the program on different source files (which should at least include the 16 `CS316exk.java` files):

```
    java -cp .   TJasn.TJ   TinyJ-source-file-name   output-file-name
```

For example: `java -cp .   TJasn.TJ   CS316ex12.java   12.out`
[Your working directory should be your *home directory* on venus or euclid, and `c:\316java` on a PC.]

When it asks you `Want debugging stop or post-execution dump? (y/n)`

you should enter `y`

When it then asks

```
Enter MINIMUM no. of instructions to execute before debugging stop.
(Enter -1 to get a post-excution dump but no debugging stop.):
```

you should enter `0`

When it then asks

```
Stop after executing what instruction? (e.g., PUSHNUM)
(Enter * to stop after executing just 0 instructions.):
```

you should enter `*`

You can execute ***my solution*** to Assignments 2 and 3 as follows:

```
 java -cp TJsolclasses:.    TJasn.TJ  CS316exk.java  k.sol [venus or euclid]
 java -cp "TJsolclasses;."  TJasn.TJ  CS316exk.java  k.sol [PC, working dir=c:\316java]
```

Here `CS316exk.java` is the TinyJ source file and `k.sol` is the output file. Compare the output files produced by your and my solutions, at least for the 16 `CS316exk.java` files; you can compare files using `diff -c` on venus and euclid, or `fc.exe /n` on your PC. *When the input file is a correct TinyJ program, the output files produced by your and my solutions need not be identical* ***but must contain*** ***the same "Instructions Generated:" lists*** ***(near the end, just above the "Data memory dump")***.

How to Submit Your Solution*

This assignment counts **2%** towards your grade if the grade is computed using rule A. To submit:

1. **Add a comment at the beginning of** `ParserAndTranslator.java` **that states your name and the names of the students you worked with (if any). As before, you may work with up to two other students, but see the remarks about this on p. 3 of the first-day announcements.**
2. **Leave your final version of** `ParserAndTranslator.java` **in your** `TJasn` **directory on euclid, so it replaces the original version of that file, 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 venus or your own PC, you can transfer the file** `ParserAndTranslator.java` **to euclid by following the instructions on p. 5 of the TinyJ Assignment 1 document; but you should substitute** `TJasn/ParserAndTranslator.java` **for** `TJ1asn/Parser.java` **and substitute** `TJasn` **for** `TJ1asn` **when following the instructions.)**

Be sure to test your submission on euclid. If your modified version of `ParserAndTranslator.java` cannot even be compiled without error on ***euclid***, then you will receive no credit for your submission. **Do NOT open your submitted file `ParserAndTranslator.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. (It's OK to view the file using the `less` file viewer after the due date.)**

As stated on page 3 of the first-day announcements document, you are required to keep a backup copy of your submitted file on venus, and another copy elsewhere. You can enter the following two commands on **euclid** to email a copy of your submitted file to yourself and to put a copy of the file on **venus**:

```
   echo . | mailx -s "copy of submission" -a TJasn/ParserAndTranslator.java  $USER
   scp  TJasn/ParserAndTranslator.java  your venus username@mars.cs.qc.cuny.edu:
```

*If euclid unexpectedly goes down after 6 p.m. on the due date, the deadline will ***not*** be extended.