Project 10 and 11 - Jack Compiler

The goal of the assignment is to write a compiler to translate JACK programming language to VM code. The assignment has to be submitted on Moodle by **13 November 2019, 11.55 PM**. The viva for the same will be conducted on **16 November**, timings of which will be intimated later.

Input to the compiler

Your program will take (N+1) command-line arguments, where N is the number of input .jack files. The details of the rest are as follows.

1. The first argument denotes the number of .jack source files to be translated (N).
2. It is followed by list of N filenames, with .jack extension

Example:
./compiler 3 Main.jack Class1.jack Class2.jack

For each .jack file, two files have to be generated - .vm file and .err file (if errors are present). For example, the VM code for Main.jack is written into **Main.vm**, and if any errors are encountered during the translation, they are printed into **Main.err**.
Only correctness or error handling is checked for each test case, ie. each test case, if error-free, should have correct VM code or if it has errors, it should have correct .err file output. The output format for error is described below.

General guidelines

1. Single line and multiline comments have to be handled.
2. Make your code as much modular as possible with appropriate comments on working of each block.
3. General coding guidelines are expected to be followed - meaningful variable names, indentation, etc.
4. The assignment is **really long**, please start early to finish early. No extensions are possible as it is the end of the semester.

Error handling

At any point, it is sufficient to stop at the first error.

Two types of error handling have to be done in the compiler.

1. Syntax error
If type matches, print
`ERROR: <TOKEN>`

If type mismatch, print
`ERROR: Expecting <TYPE> but <TOKEN>`

Example:

```
var int foo;
let foo = 2
let foo = foo + 1;
```

should print the error **`ERROR: Expecting <symbol> but let`**

2. Undeclared variables
Any variable used without declaration should be reported as an error.
`Declaration error: <varname> undeclared.`

Example 1:
var int a, b, c;
let c = a + 1;
let d = a - b; // Error #1

should print the error Declaration error: **`Declaration error: d undeclared.`**

`Example 2:`
var int a, b, c;
let c = a + 1;
let a = a - d; // Error #2

should print the error Declaration error: **`Declaration error: d undeclared.`**

Automation Script

A bash script `run_compiler.sh` is provided with instructions. Fill command to **compile your code** and **run your code**. You can run the script in the following format.

```
./run_compiler.sh 2 Main.jack Class1.jack
```

Please note that the command line arguments given to the bash script **are exactly the same** as that is passed to your compiler program.

In case of any doubts, please ask in Moodle.

Submission Format

You should submit a single tar.gz file with the name of your roll number. When extracted, it should create a folder with your roll number as name. Inside the folder, you can place all your code. **Please do not forget to add your run_compiler.sh to the folder.** For example, if your roll number is CS15B033, create a folder named CS15B033, put all your source files and run_compile.sh inside it, compress it to CS15B033.tar.gz and the same should be submitted on Moodle.