170025298， Michael Weir， 16/10/2018
W05 FSM

1. Design:
The following are logics when programming the program:

1.1 FSM File Operation:
read file → read lines → split words → save result

1.2 Input argument Operation:
Split input → save as arrays

1.3 FSM Operation: get initial statement:
statement1 → check input → output output → check next statement → {next statement → repeat, this statement → statement 1; if not available throw bad input} → final result

1.4 Bad description design:
1.4.1 Every current state contains equal solution to every input:
create a tree set → add current input to the set → get size of the set → check each state contains size of the set of solutions → if not throw exception
Problem happens to the design: During using for loop to read across sets, String always appeared with null but if I only read across the sets, no null value appeared. To prevent this I uses

```
if(!contentsMap.containsKey(o + u) && o != null && u != null) {
    throw new BadDescriptionException();
}
```

However I don't understand why o and u have null value and they shouldn't happens.

1.4.2 max next state is less than max current state
from FSMProcess method, get max current state and max next state → if max state is not equal → throw new Exception.

2. Test:

2.1 Java Junit Test:
2.1.1 read file test: This checks files can be read and export as a list.
2.1.2 scanner test & 2.1.3 saving data test: This checks input String can be splitted into four parts and save as an FSMContent class type.
2.1.4 input test: This tests input stream can be diveded as a single character and makes it easy for program to read.
2.1.5 content test: This tests all data in FSMcontent class can be retrieved successfully.
2.1.6 client test: This tests that client is not null.
2.1.7 Client input test: this tests input test works well in client and input stream class.
2.1.8 FSMFile test: This tests that FSM process runs normally.
2.1.9 Bad description check: This checks method of check fsm file works well so that it fits to the design part(1.4).

2.2 Stacscheck Test:

```
pc5-003-l:~/Documents/CS2001/W05-FSM/src hl74$ stacscheck /cs/studres/CS2001/Pra
cticals/W05-FSM/Tests/
Testing CS2001 Week 5 Practical (FSM)
- Looking for submission in a directory called 'src': Already in it!
* BUILD TEST - basic/build : pass
* COMPARISON TEST - basic/Test01_simple/progRun-expected.out : pass
* COMPARISON TEST - basic/Test02_bigger/progRun-expected.out : pass
* COMPARISON TEST - basic/Test03_description/progRun-expected.out : pass
* COMPARISON TEST - basic/Test04_missinginput/progRun-expected.out : pass
* COMPARISON TEST - basic/Test05_illegal/progRun-expected.out : pass
* COMPARISON TEST - basic/Test06_minimal/progRun-expected.out : pass
* COMPARISON TEST - basic/Test07_alsodescription/progRun-expected.out : pass
* COMPARISON TEST - basic/Test08_sensible/progRun-expected.out : fail
--- expected output ---
13ff277301
--- submission output ---
Bad input
---

8 out of 9 tests passed
```

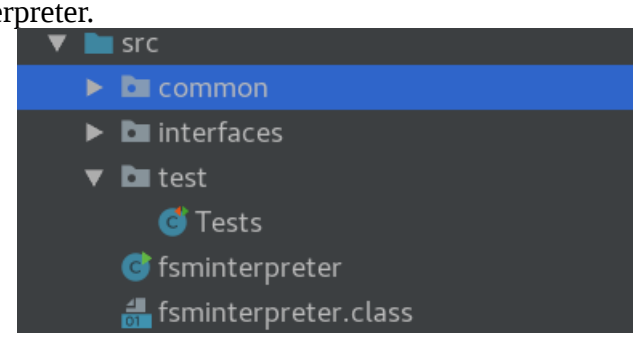
8/9 passed, however for the final test I don't know where mistakes are, which leads to the confusion of my program. However the rest of test works well.

```
Testing CS2001 Week 5 Practical (FSM)
- Looking for submission in a directory called 'src': Already in it!
* BUILD TEST - basic/build : pass
* COMPARISON TEST - basic/Test01_simple/progRun-expected.out : pass
* COMPARISON TEST - basic/Test02_bigger/progRun-expected.out : pass
* COMPARISON TEST - basic/Test03_description/progRun-expected.out : pass
* COMPARISON TEST - basic/Test04_missinginput/progRun-expected.out : pass
* COMPARISON TEST - basic/Test05_illegal/progRun-expected.out : pass
* COMPARISON TEST - basic/Test06_minimal/progRun-expected.out : pass
* COMPARISON TEST - basic/Test07_alsodescription/progRun-expected.out : pass
* COMPARISON TEST - basic/Test08_sensible/progRun-expected.out : fail
--- expected output ---
13ff277301
--- submission output ---
13ff177123
---

8 out of 9 tests passed
```

After reading the file of 8th test I found that statements are with alphabet order so I changed into set first state as the top line in the fsm file, however failed tests still available.

What's more, as I uses interfaces and packages to program the program, the stacscheck are unable to compile all java programs in the src file, so I have compiled al programs and then stacscheck always only read fsminterpreter.



3. Extensions:
3.1 Bad Description Extension: This makes program are able to run when the fsm file contains not all of the current input state such as:
1 1 O 2
1 2 O 4
2 1 E 1
2 2 O 3
3 1 E 4
4 1 O 3
4 2 E 1
and one line lost in this fsm file.
However when such problem happens, the program will return as <null> to prevent such problem such as

```
/usr/lib/jvm/java-1.8.0-openjdk/bin/java ...
12121212121211212
OOEEOOEEOOEEOEOO<null>
Process finished with exit code 0
```

3.2 Extension 2 allows input being putted separately such as

```
12111
OOEOE
1211111
O<null>EOEOE
1211111111
O<null>EOEOEOEO
121121212121
EEOEOO<null>EEOOE
1212121321
O<null>EEOOEBad input

Process finished with exit code 1
```