Documentation:
Name: Rodolfo Anguiano
ID: 915813033
Link to github: https://github.com/SFSU-CSC-413/assignment-2-royanguiano
Instructions:
The purpose of this project is to add three additional types of tokens to the lexer.
We will be adding the following tokens: >, void, float.
Tokesetup takes care of setting up the tokens from file into the hashmap and creates a new type of token.
The token.java was modified to print out the line numbers in the output.
I have also modified the output format with string.format to match requirement #4 in assigment.
Summary of technical work:
When creating new types of tokens please include the following types of characters or tokens in the simple.x
file for the sourceReader to read in and create them in the hasmap.

Excution and Development environment described:
executing the projects takes a couple steps.
1. setup the token by adding new types of tokens to simple.x file
2. run sourceReader class to create the tokens.
3. run lexer class to print out formated strings.

scope of work described:
Instructions to compile and execute jar:
Instructions for jar file:
Start Command Prompt.
Navigate to the folder that holds your class files:

```
C:\>cd \mywork
```

Set path to include JDK’s bin. For example:

```
C:\mywork> path c:\Program Files\Java\jdk1.7.0_25\bin;%path%
```

```
Compile your class(es):
C:\mywork> javac *.java
Create a manifest file and your jar file:
C:\mywork> echo Main-Class: Craps >manifest.txt
C:\mywork> jar cvfm Craps.jar manifest.txt *.class
or
C:\mywork> jar cvfe Craps.jar Craps *.class
Test your jar:
C:\mywork> Craps.jar
or
C:\mywork> java -jar Craps.jar
```

Assumptions made:
I didnt know how the lexer worked. I assumed that it was a character array and it breaks down the ar ray into
seperate strings and characters.

Class diagram with hierarchy:
Implementation decisions: Check attachement
Code organization:
Results/Conclusion:

```
 //output:
Source file: simple.x
user.dir: C:\Users\roy\IdeaProjects\assignment-2-ro
yanguiano
READLINE:   program { int i int j
program    left:   0     right:   6 line:   1
{          left:   8     right:   8 line:   1
int        left:  10     right:  12 line:   1
i          left:  14     right:  14 line:   1
int        left:  16     right:  18 line:   1
j          left:  20     right:  20 line:   1
READLINE:      i = i + j + 7
i          left:   3     right:   3 line:   2
=          left:   5     right:   5 line:   2
i          left:   7     right:   7 line:   2
+          left:   9     right:   9 line:   2
j          left:  11     right:  11 line:   2
+          left:  13     right:  13 line:   2
7          left:  15     right:  15 line:   2
READLINE:      j = write(i)
j           left:   3     right:   3 line:   3
=           left:   5     right:   5 line:   3
write       left:   7     right:  11 line:   3
(           left:  12     right:  12 line:   3
i           left:  13     right:  13 line:   3
)           left:  14     right:  14 line:   3
READLINE:   }
}           left:   0     right:   0 line:   4
1. program { int i int j
2.     i = i + j + 7
3.     j = write(i)
4. }


Process finished with exit code 0
```