Assignment 1

Implement a scanner for the programming language with the following lexical structure.

```
comment ::=   /*   NOT(*/)*  */
token ::= ident  | keyword | frame_op_keyword | filter_op_keyword | image_op_keyword | boolean_literal
          | int_literal  | separator  | operator
ident ::= ident_start  ident_part*    (but not reserved)
ident_start ::=  A .. Z | a .. z | $ | _
ident_part ::= ident_start | ( 0 .. 9 )
int_literal ::= 0  |  (1..9) (0..9)*
keyword ::= integer | boolean | image | url | file | frame | while | if | sleep | screenheight | screenwidth
filter_op_keyword :: = gray | convolve | blur | scale
image_op_keyword :: = width | height
frame_op_keyword :: = xloc | yloc | hide | show | move
boolean_literal ::= true | false
separator ::=     ;  | ,  |  (  |  )  | { | }
operator ::=      |  | &  |  ==  | !=  | < |  > | <= | >= | +  |  -  |  *   |  /   |  %  | !  | -> |  |-> | <-
```

- If an illegal character is encountered, your scanner should throw an IllegalCharException. The message should contain useful information about the error. The contents of the message will not be graded, but you will appreciate it later if they are helpful.
- If an integer literal is provided that is out of the range of a Java int, then your scanner should throw an IllegalNumberException. The contents of the message will not be graded, but you will appreciate it later if they are helpful.

Additional requirements:

- This code must remain in package cop5556sp17(case sensitive): do not create additional packages.
- Names (of classes, method, variables, etc.) in starter code must not be changed.
- Unless otherwise specified, your code should not import any classes other than those from the standard Java distribution.

Comments and suggestions:

- The given scanner should compile correctly with the junit test. When executed, only one test will pass, but all should pass in your completed scanner.
- Work incrementally: add a capability along with a junit test to exercise it incrementally
- You will probably want to develop some methods to encapsulate checks to make it easier to write JUnit test cases.
- If you use Integer.parseInt to get the value of a numeric literal, it will throw a NumberFormatException if the value is too large. This is useful functionality, but the

exception is not the same one as specified. You need to catch it and throw a Scanner.IllegalNumberException with a useful message.