CSC 345 Lab – PushbackReader

Overview

Write a Java program that uses a PushbackReader.

Create Project

Create an IntelliJ console application. Make sure to use Maven as the build system when creating the project.

Create Input File

Create an input file named "input.txt" with the follow data inside of it:
abc: de, "fg"h123 ijk 456

Use PushbackReader

Write code in main that uses a PushbackReader to read data from a file and display it on the screen. The PushbackReader should use a FileReader as its data source. When setting up the FileReader it should be connected to "input.txt". Create a loop that reads and prints all data from the file. Here is sample output:

```
Start reading from file
a
b
c
:

d
e
,

"
f
g
"
h
1
2
3

i
j
k
```

4
5
6
Done reading from file

Show Messages for Each Character Category

Update the code so that it prints a message depending on the category of the character.

Hint: There are static methods in the Character class that can help with checking if a character is a letter or digit.

Character	**Message**
Character is a letter	Letter
Character is a digit	Digit
Blank space	Space
:	Colon
,	Comma
“	Double quote

Here is sample output:

Start reading from file
a Letter
b Letter
c Letter
: Colon
  Space
d Letter
e Letter
, Comma
  Space
" Double quote
f Letter
g Letter
" Double quote
h Letter
1 Digit
2 Digit

```
3 Digit
  Space
i Letter
j Letter
k Letter
  Space
4 Digit
5 Digit
6 Digit
Done reading from file
```

Show Message for Strings

Update the code so that it prints the message String if it encounters a string of characters. It should no longer be printing the Letter message. It should print the message String once for each string of characters it encounters. Here are some hints:

- You will need to add a nested loop inside the main loop that processes the file.
- If a letter is encountered, it should enter this nested loop. This loop should keep going until it reads a character that is not a letter.
- After leaving the nested loop, it should push the character that caused it to break out of the nested loop back into the input stream.

Here is sample output (the output on each String line shows the character that appears after the string):

```
Start reading from file
: String
: Colon
  Space
, String
, Comma
  Space
" Double quote
" String
" Double quote
1 String
1 Digit
2 Digit
3 Digit
  Space
  String
  Space
4 Digit
5 Digit
6 Digit
Done reading from file
```

Show Message for Numbers

Update the code so that it prints the message Number for strings of digits. This is like what was done with strings.

Here is sample output (the output on each Number line shows the character that appears after the number):

```
Start reading from file
: String
: Colon
  Space
, String
, Comma
  Space
" Double quote
" String
" Double quote
1 String
  Number
  Space
  String
  Space
  Number
Done reading from file
```

Buffer the Actual Strings and Numbers

Update the code so that it adds string and number characters to a buffer. Use an instance of StringBuilder as the buffer. Display the contents of the buffer (the StringBuilder) at the end of each line.

Here is sample output (the data from the buffer appears last on each line):

```
Start reading from file

: String  buffer: abc
: Colon  buffer: :
  Space  buffer:
, String  buffer: de
, Comma  buffer: ,
  Space  buffer:
" Double quote  buffer: "
" String  buffer: fg
" Double quote  buffer: "
1 String  buffer: h
```

```
  Number  buffer: 123
  Space  buffer:
  String  buffer: ijk
  Space  buffer:
  Number
Done reading from file
```