TCSS143
Fundamentals of Object-Oriented Programming-Theory and Application
Programming Assignment 8
Or
A project for all of us who find ourselves frequently out of the loop…
Recursion!

The purpose of this programming project is to demonstrate the functionality of Recursion while reviewing such things as the use of Sets (here only a single Set will be implemented), String manipulation, file I/O using the Scanner class for the input file.

REQUIREMENTS

Though there will be several instances where iteration would solve the problem, **Absolutely NO loops should be found in any of your code!** It is a requirement of this assignment that **all** repeated processes will be performed through recursion.

You will submit a single file "**Assignment8.java**" **(NO Zipped File)** through the Programming Assignment 8 Submission link on Canvas. This file will be your solution to this assignment. Too, you can use the file I posted on Canvas or create your own test input file "In8.txt" **BUT**, you will **NOT** include this in8.txt in the upload to Canvas. I will use my own for testing.

Not only will you be graded on program correctness (Program executes correctly, proper use of methods, classes, inheritance, etc.) but also, good programming documentation techniques as we have learned during this course.

1. Assignment8.java

All methods used to solve this problem will be contained in a single file that includes main. Of course, these methods will be declared as static.

At very least, you are to write 3 separate recursive methods to solve this problem. Their intent is as follows:

> **getWordsString**: Recursively read all the words contained in the input file and return a String containing only the words found in the file that contain a given char (The words within this String will be separated by a space). Multiple return statements will be allowed here.
>
> **hasCharacter**: Recursively scan the characters within a received word to see if it contains a given char and return true if it does, false otherwise.
>
> **getWordSet**: Recursively scan the words in a received String where each word is separated by a space and placing each word into a Set (HashSet) which is returned to the calling program once all the words have been scanned. Multiple return statements will be allowed here.

Other details of each of these methods follows the discussion on main below.

main:

This program is to read all the words contained in the input file, one at a time recursively until no words remain. To do this main will call a recursive method named getWordsString (discussed shortly) which is passed a Scanner to a File object and a single character. getWordsString will read all the words in the file and return a String containing all words from the input file that contain a single given character. Keep in mind, this string must have a single space inserted between each word.

In order to remove redundant words, the String returned from getWordsString will be sent to another recursive method "getWordSet" which will 'pick' out each word in the String and add them to a HashSet (this will remove all duplicates). The HashSet will be returned to main for final output. Details on getWordSet are listed below.

main will output this Set to out8.txt (No special formatting here. Just print the Set to the output file "out8.txt" with a single println statement).

This program MUST read a file named in8.txt and generate an output file named out8.txt. The in8.txt file must be created by you based on formatting described shortly. In8.txt and out8.txt are **NOT** to be included during the submission process.

public static String getWordsString(Scanner theFile, char theC)
The method getWordsString is passed the open input file through a Scanner and a single char (your choice) and returns a String of words (each separated by a space) found in the file that contain the char argument. This method will **recursively** read each word contained in the input file. As each word is read, you will check to determine if it contains the given character through a call to the method hasCharacter discussed below

This means you can only use length, indexOf, substring, and charAt. You are not allowed to use any other existing String methods or other pre-written methods to determine if the char is in the String. You have to write this process.

If the current word contains the given char, then this word should be concatenated onto a String along with a space to separate each word. This will be tied in with the return statement that includes the recursive call.

When all words in the file have been processed, getWordsString will return this String of words that contain the given character.

Multiple return statements will be allowed here.

public static Boolean hasCharacter(String theS, char theC)
The hasCharacter method will recursively check the first character of the continuously smaller substring for the given character until the length of the substring reduces to 0, at which point no match was found and the method should return false. However, if the length is > 0 and a match is found, hasCharacter should return true. This means there are 2 base cases here. For all other cases, the length is > 0 AND the character is not found, you should return the result of the recursive call.

You can use the charAt method here. Also, though the character being used for testing is lower case, don't forget to check for the uppercase equivalent. toUpperCase or any other pre-written methods are NOT allowed. You have to make this conversion yourself (ASCII 'a' is 32 greater than 'A', 'b' is 32 greater than 'B;, and so on).

public static Set<String> getWordSet(String theS)
This method will be called by main following the call to getWordsString. It receives the String created by getWordsString and returns a Set (using a HashSet) of the words contained in the String theS. The process should be done recursively with each call passing a substring of removing one word (each removed words is added to the set).

You are allowed to use the String methods: indexOf, and substring

And the Set methods: add and addAll

Any other method calls appearing in this method must be written by you.

Multiple return statements will be allowed here.

In8.txt

The input file "In8.txt" will contain a series of words. This can be a poem, a set of instructions, an email message, etc., as long as it's a text file. See the sample run at the end of this assignment for the in8.txt file I provided.

Out8.txt

All output (the above mentioned) should be sent to the output file out8.txt. See sample run for example.

As a suggestion, complete this project in steps:

- ✓ Write main to open read and write your I/O files (just echo input to output).
- ✓ Write and call from main, the getWordsString method and output the resulting String. At this stage, have getWordsString call a simple version of hasCharacter that checks for the given character any way you see fit and returns the expected true of false result.
- ✓ Run this for testing and any corrections that need to be made.
- ✓ Re-write the code inside hasCharacter to follow the specifications above in the description. Remember, no loops or outside helper methods. You can only use length, charAt, and substring.
- ✓ Run this for testing and any corrections that need to be made.
- ✓ Write the getWordSet method, test and correct as needed.

Also, just prior to your final output of the Set to the file, you might want to execute a single remove an empty string (2 double quotes with nothing in between) operation on the Set.

Remember, absolutely no loops!

If you feel the need, please implement any other methods you think the above classes may need.

Sample I/O may appear as follows:

Suppose In8.txt contains:

Astronomy Domine (Barrett) 8:31

Lime and limpid green, a second scene
A fight between the blue you once knew.
Floating down, the sound resounds
Around the icy waters underground.
Jupiter and Saturn, Oberon, Miranda and Titania.
Neptune, Titan, Stars can frighten.

Blinding signs flap,
Flicker, flicker, flicker blam. Pow, pow.
Stairway scare Dan dare who's there?

Lime and limpid green, the sounds surronds
The icy waters under
Lime and limpid green, the sounds surronds
The icy waters underground.

Then the output should contain:

[Dan, can, (Barrett), A, Stars, a, Miranda, Astronomy, Floating, flap,, and, Around, Stairway, waters, blam., Titan,, Titania., Saturn,, scare, dare]

In this example you might note the occasional extra commas (,). This is because the input included commas and are considered part of the word by the next( ) method.

No need to zip anything (and please do not). Just submit your single Programming8.java file.

The next and final page includes an outlined overview of this assignment -->

Addendum to the initial Programming Assignment 8 document

This document is to be "In Addition To" the original Programming Assignment 8 and is intended to describe each method in a clearer, outline approach. All requirements in Programming Assignment 8 remain intact and take precedence over all else (should not be ignored). Also, you should refer to the initial document for specifics (don't forget, No Loops Allowed and you are only allowed to use the String methods: indexOf, substring, length, and charAt. No other methods but your own can be used or applied to the String(s)):

main (String[] theArgs):

- Creates a Scanner to an input file: in8.txt.
- Creates a PrintStream to an output file: out8.txt.
- Calls getWordsString and passes the input file scanner & the single character 'a' or any of the 52 characters:
    - getWordsString returns a String made of all words (separated by space)in the input file that contain the letter 'a'.
- Passes the String returned from getWordsString to getWordSet which returns a HashSet containing all the words in the String.
- Removes the empty String from the HashSet (if it exists)
- Outputs the HashSet to the output file: out8.txt.

public static String getWordsString(Scanner theFile, char theC)

- Recursively reads all words in theFile while concatenating into a String, only words that contain the letter in theC.
- Calls hasCharacter (passing the current word and char theC) to determine if the word should be concatenated into the String (if so, the word, a space, and the return String from a recursive call to getWordsString is returned. Otherwise, only the return value of a recursive call to getWordsString is returned.
- (Base Case) An empty String is returned when no words remain in theFile.
- Needs a local String variable for each word read.

public static boolean hasCharacter(String theS, char theC)

- Recursively (reduces the String theS by 1) scans the given String checking in each call for a match with char theC. (Each of these recursive calls are returned to the previous call. This is the general or recursive case)
- (Base Case)Returns true if char theC is found (need to check both lower and upper case in the String)
- (Base Case)Returns false if the String length is < 1.
- (Upper case char's are the int value of 32 less than their lower case equivalent)

public static Set<String> getWordSet(String theS)

- Receives the String of words each of which is terminated with a space (last word will likely be followed by a space).
- (General or Recursive Case) Adds a word from the String to a local Set and Recursively calls itself with a reduction of the String by 1 word. This call will return a set who's "All" elements are to be "Added" to the local Set. The local set is then returned.
- (Base Case) Only 1 word in the String which is added to the local set and the local Set is returned.