# Count the Number of Words in a Poem

**Learning objective:**
Through this activity, you will practice writing and using static methods, reading text input through a Scanner class, implementing String methods, and using a loop to iterate through data

**Problem Description:**
Your friend wants to know how many times a certain word pops up in his favorite poem, but he is too lazy to count by himself. Thus, he has asked you to help him write a program to do it for him! Your program will take in a text file through the command line. It will first read the file and print its lines. It will then take a word input from the user and count how many times that word occurs throughout the text.

**Provided Files:**
- PoemReader.java - This is a skeleton of the class that you will need to write
- poem.txt - This is an example poem for you to text your code. It should be in the same directory as PoemReader.java.

**Useful APIs**
This assignment relies heavily on the following classes. We highly recommend you reference the Java SE 8 API for the following classes if you have any questions regarding their functionality and/or associated methods
- Scanner - Class for parsing text based input
- String - Represents character strings
- File - Abstract representation of a file/directories

**Solution Description**:

**Part 1 :**
Your objective for this part is to print out all the lines from a file you read in verbatim and in the same format.

1) Fill in the function printFile that takes in a File and prints out all of the lines. The output should be in the same format as if you opened the file in a text editor
2) Fill in the main method so that it can take in file path from System.in (using Scanner). Create a File object using this path and pass in this file to printFile so it can be printed out.

Note:
- Only pass the File object to the printFile function, don't pass the string of where the file is located.
- Although there are many ways to read a File, using a Scanner is recommended

- printFile should not return anything, do all of the printing inside the function itself
- Make sure the file you enter exists and is in the correct folder
- Watch out for formatting issues that will make the text unreadable. For example, make sure you don't print the entire poem out on one line.
- Any time you ask the user for input, make sure to ***prompt*** them first:
    - i.e. make sure you give them some warning they need to input something before asking for an input, such as :
    - System.out.print("Please enter a file path: ");

**Part 2 :**

Now that you've managed to print out the file properly, it is time to count the words!

1) Fill in the function wordCount that takes in a File and a String. The function returns how many times that String appears in the file. The String that is passed in should only ever contain one word.
2) Update the main method such that:
    a) A word to look for in wordCount is taken from System.in. Save this input and use it for your wordCount function. (Hint: Use the Scanner class!)
    b) Use the wordCount function to find out how many times the word exists in the file.
    c) Print out the number of occurrences of the word in the following format:
        i) "The word _____ appears ____ times", where the blanks are the word and the number of times it appears respectively

Notes:

- You should account for capitalization issues.
    - For example, "Hello" should be counted if the user entered "hello"
    - Hint: look at the String API to see if there are useful functions for this
- There may be more than one occurrence of a word in a line.
    - For example, "The fruit vendor was out in the street selling fruit." Contains two instances of the word "Fruit"
- Punctuation may be removed through the line
    - String textNoPunct = text.replaceAll("\\p{Punct}", "");
    - You do not need to understand why this works
    - Note this only works on ***Strings,*** do not try to use replaceAll on a file or any other data type that's not a String
- Make sure you account for words that are substrings of another:
    - For example, "I love applesauce" should not count the word "apple"
- Any time you ask the user for input, make sure to ***prompt*** them first:
    - i.e. make sure you give them some warning they need to input something before asking for an input, such as :
    - System.out.print("Please enter a word: ");

**CHALLENGE:**

**Part 3:**

Now that you've implemented the word counter, try to implement a function to keep track of the last line of where the word occurs!

1) Fill in the function findLastOccurrence that takes in a File and a String. The function returns the line number where that String last occurs. The String should contain only a single word.
    a) Assume the first line is line 1
    b) If the word does not appear at all, the function should return -1
2) Update the main method such that:
    a) It calls findLastOccurence with the same word that was used when calling wordCount.
    b) If the word was found, it prints out where the word last occurred in the following format:
        i) "Last occurance of _____ is in line _____", where the blanks are the word and the line in which it appears respectively
    c) If the word was not found, it prints out the following:
        i) "Your word was not found."

Notes:

- As with the last part, keep in mind capitalization, multiple occurrences, punctuations, and words that are substrings of one another
- You can reuse the code that prompts the user to input a word. You do not need to ask the user for a new word.