Assignment 4: String Shortener

Instructions

Ever tried to write a tweet that was too long to post? There's nothing particularly new about Twitter's character limit: from the very start of electronic communications, there have been benefits to reducing message length. Telegraph companies in the 19th and early 20th century charged users by the word and had a maximum number of letters per word. When SMS first allowed cellphone users to send text messages in the 1990s, technical limitations meant that a maximum of 160 characters could be used for each message. The same technical limits meant Twitter launched with a 140 character limit, which has since doubled to 280. By enforcing a character limit, Twitter can encourage people to display only the most necessary information in a concise way, allowing you to quickly scroll through a feed and see many tweets.

Whenever the length of messages is limited, people will come up with creative ways to shorten their message to fit. This includes the use of abbreviations and code phrases which are used to remove filler words and letters: for example "good" becomes "gd". See webopedia's **Huge List of Texting and Online Chat Abbreviations (https://www.webopedia.com/quick_ref/textmessageabbreviations.asp)** for a more than comprehensive list of abbreviations like this. Some people even use an automated approach (e.g. **https://tweetcompressor.com/ (https://tweetcompressor.com/)**) to substitute characters.

In this assignment, you are going to try and write two algorithms to do this job: shortening a message length by reducing the number of characters, while still maintaining the important information. This is an example of "compression" - an extremely important area of computer science where data usage is reduced. There is always a trade-off when compression is involved - saving more data can mean some important information is lost, so bear that in mind when you compare your two algorithms.

The assignment

Your job in this assignment is to write a program that takes a message as a string and reduces the number of characters it uses in two different set ways. The first thing your program will do is ask the user to type a message which will be stored as a `String`. The String entered should be immediately converted to lowercase as this will make processing much easier. You will then apply two different algorithms to shorten the data contained within the String.

Algorithm 1

This algorithm creates a string from the message in which every vowel (a, e, i, o, and u) is removed *unless* the vowel is at the very start of a word (i.e., it is preceded by a space or is the first letter of the message). Every repeated non-vowel character is also removed from the new string (i.e., if a character appears several times in a row it should only appear once at that location). So for example the string "I will arrive in Mississippi really soon" becomes "i wl arv in mssp rly sn".

After applying this algorithm, your program should output the shortened message, the number of vowels removed, the number of repeated non-vowel characters removed, and how much shorter the shortened message is than the original message. The exact format in which the program should print this information is shown in the sample runs.

Algorithm 2

This algorithm creates a string by taking each unique character in the message in the order they first appear and putting that letter and the number of times it appears in the original message into the shortened string. Your algorithm should ignore any spaces in the message, and any characters which it has already put into the shortened string. For example, the string "I will arrive in Mississippi really soon" becomes "8i1w4l2a3r1v2e2n1m5s2p1y2o".

After applying this algorithm, your program should output the shortened message, the number of *different* characters appearing, and how much shorter the shortened message is than the original message. The exact format in which the program should print this information is shown in the sample runs.

Sample Run 1

```
Type the message to be shortened
This message could be a little shorter

Algorithm 1
Vowels removed: 11
Repeats removed: 2
Algorithm 1 message: ths msg cld b a ltl shrtr
Algorithm 1 characters saved: 13

Algorithm 2
Unique characters found: 15
Algorithm 2 message: 4t2h2i4s1m5e2a1g1c2o1u3l1d1b2r
Algorithm 2 characters saved: 8
```

Sample Run 2

```
Type the message to be shortened
I will arrive in Mississippi really soon

Algorithm 1
Vowels removed: 11
```

```
Repeats removed: 6
Algorithm 1 message: i wl arv in mssp rly sn
Algorithm 1 characters saved: 17

Algorithm 2
Unique characters found: 13
Algorithm 2 message: 8i1w4l2a3r1v2e2n1m5s2p1y2o
Algorithm 2 characters saved: 14
```

Milestones

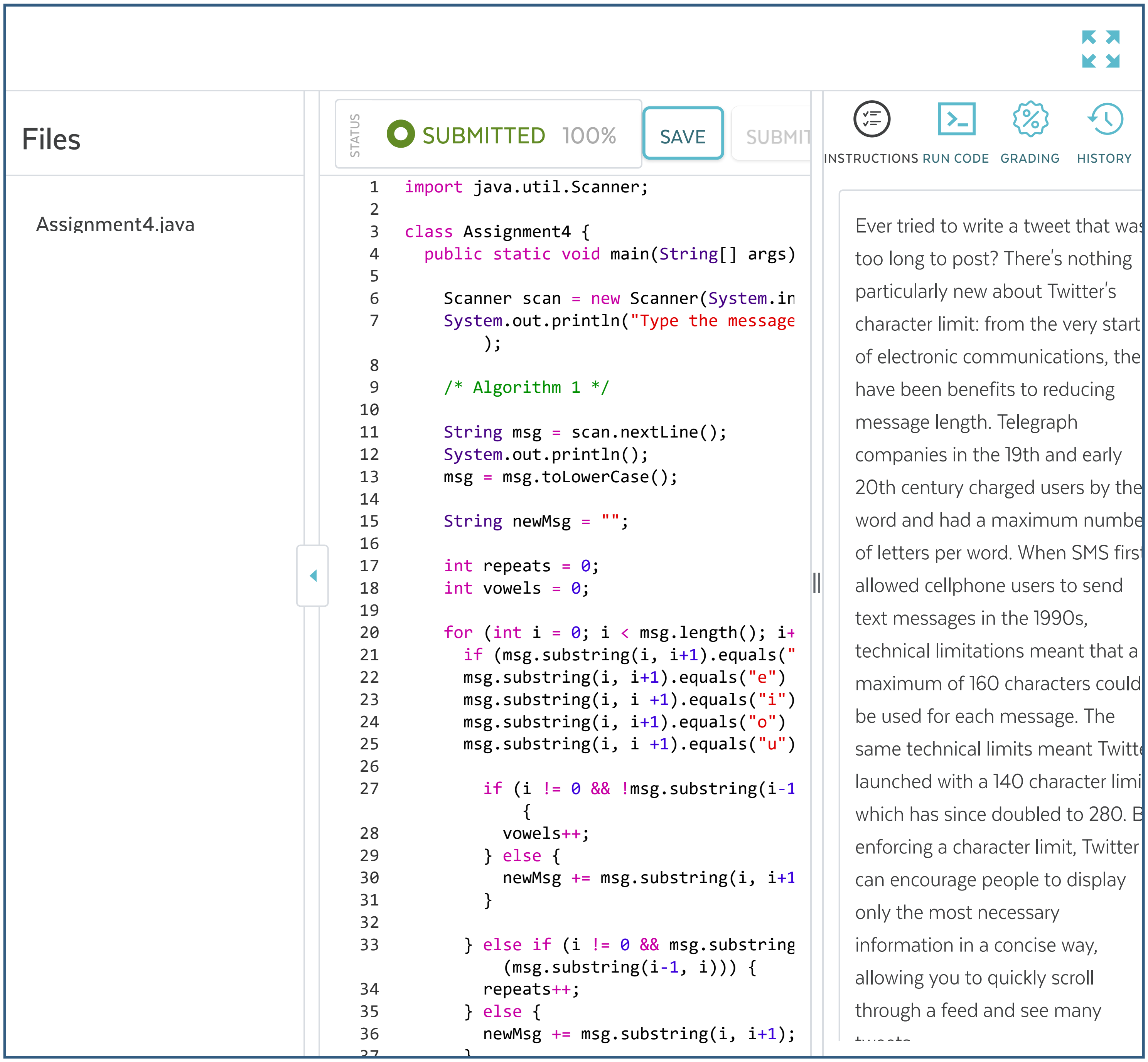
As you work on this assignment, you can use the milestones below to inform your development process:

Milestone 1: Set up a program that takes a string input and converts all the letters to lowercase. Start implementing algorithm 1: create a counter variable and iterate through the characters of the String, incrementing this each time a vowel is encountered which is not preceded by a space or is at the start of the String. So at the end of the loop this counts the number of vowels that are not at the start of a word.

Milestone 2: Add further conditions (using else if) in your loop to count any non-vowel characters which appear immediately after the same character. Make a new empty String to hold the shortened message at the start of the code, then add a final else condition in the loop to add all characters which were not vowels or repeated letters to this String. Then print the statements for algorithm 1 using your counts and shortened message.

Milestone 3: Start implementing algorithm 2 by writing code that iterates through the String and checks that each character is not a space and has not already appeared in the word before that point. You will need to use nested loops - an outer loop to iterate through the String characters and an inner loop that looks through the previous characters up to that point - and a flag variable to record if a letter was found in the inner loop. Use a counter variable to count all such "unique" characters in the String.

Milestone 4: Add a second inner loop inside the outer loop from the previous which counts all appearances of a character that passes the tests from milestone 3. Add the character and the number of times it appears to another shortened message String (which should start as blank String). Finally, print the statements for algorithm 2 using your unique character count and shortened message.



```
import java.util.Scanner;

class Assignment4 {
  public static void main(String[] args)

    Scanner scan = new Scanner(System.in
    System.out.println("Type the message
        );

    /* Algorithm 1 */

    String msg = scan.nextLine();
    System.out.println();
    msg = msg.toLowerCase();

    String newMsg = "";

    int repeats = 0;
    int vowels = 0;

    for (int i = 0; i < msg.length(); i+
      if (msg.substring(i, i+1).equals("
      msg.substring(i, i+1).equals("e")
      msg.substring(i, i +1).equals("i")
      msg.substring(i, i+1).equals("o")
      msg.substring(i, i +1).equals("u")

        if (i != 0 && !msg.substring(i-1
            {
          vowels++;
        } else {
          newMsg += msg.substring(i, i+1
        }

      } else if (i != 0 && msg.substring
          (msg.substring(i-1, i))) {
        repeats++;
      } else {
        newMsg += msg.substring(i, i+1);
```

Ever tried to write a tweet that was
too long to post? There's nothing
particularly new about Twitter's
character limit: from the very start
of electronic communications, the
have been benefits to reducing
message length. Telegraph
companies in the 19th and early
20th century charged users by the
word and had a maximum numbe
of letters per word. When SMS firs
allowed cellphone users to send
text messages in the 1990s,
technical limitations meant that a
maximum of 160 characters could
be used for each message. The
same technical limits meant Twitte
launched with a 140 character limi
which has since doubled to 280. B
enforcing a character limit, Twitter
can encourage people to display
only the most necessary
information in a concise way,
allowing you to quickly scroll
through a feed and see many