Lab 03 – Parsing Tweets with the String Class

Introduction

Twitter and similar micro-blogging platforms allow users to broadcast short messages to a potentially large audience. With Twitter, the messages are called "tweets" and consist of 140 characters or less. Tweets are often sent from mobile phones, and tweeters can choose whether their tweets are public or not.

The platforms allow people to communicate in ways that were not possible even a few years ago, and they have already had a profound effect on society. They even affect how society reacts to natural and manmade disasters—ordinary citizens can now broadcast timely and location specific information that is of vital importance to both emergency management agencies and members of the public.

Sifting through the tweets for useful information is difficult, however. As a result, there have been proposals to manually add structure to tweets sent during disasters. One proposal is called Tweak the Tweet (TtT).[1] In TtT, information is marked using a "hashtag" such as **`#loc`** (for location):

```
MT @carlseelye: #lovelandfire #Wind just switched and now the smoke is
thick around our house #loc 40.352,-105.2045
```

Adding the structure makes it far easier for a computer to classify the tweets.

In this lab, you will use methods of the **`String`** class to process messages similar to TtT tweets (the data is made from real TtT data but it has been altered to suit the lab). You will use the **`substring`** and other methods to pull out information from the text, manipulate it, and print it out to the screen.

Lab Objectives

By the end of the lab, you should:

- understand how **`String`** constants and String objects are represented in Java;
- be able to declare, initialize, and assign variables of type **`String`**;
- perform string processing using the methods of the **`String`** class;
- concatenate strings (and other values) together using the '**+**' operator.

Prerequisites

The lab deals with material from Chapter 2 (specifically, the discussions of the **`String`** class). It assumes you know how define simple classes in Java, and that you can declare and assign values to variables.

Exercise – Parsing Tweets

The class **`ParseTheTweet`** will read in a single tweet from the keyboard, and so when writing your program, you will need to use the **`Scanner`** class in addition to the **`String`** class. As usual, almost everything you write will go in the **`main`** method of the **`ParseTheTweet`** class.

The tweets processed by the call all encode the following information using so-called "hashtags": The report *type* (**`#typ`**); some further *detail* (**`#det`**); a location (**`#loc`**) such as a street address; and latitude (**`#lat`**) and longitude (**`#lng`**). The type indicates the meaning of the tweet (i.e., whether it is a request for help or reports factual information), and the report detail provides additional information. Each of the hashtags is followed by some value (such as an actual latitude or longitude value).

[1] http://epic.cs.colorado.edu/?page_id=11

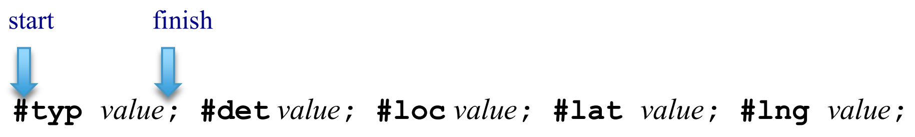
When writing your code, you can assume that all of the input tweets to process have the following format:

#typ *value*; **#det** *value*; **#loc** *value*; **#lat** *value*; **#lng** *value*;

That is, they consist of a series of hashtags, each followed by a value, and each value followed by a semicolon (;). 9 sample tweets are provided in examples at the end of this document. You should test your code on them.

Instructions

1. At the top of your source file, include a comment stating your Java class name, your name, the date, the program purpose, and containing the statement of academic honesty as you did in previous labs.
2. Prompt the user "Enter a tweet below" as shown in the examples in this document. Use the **Scanner** class (as discussed in lecture) to read in a tweet entered by the user and store it in a **String** variable named **tweet**. The program must read in an input tweet from a single line of text.
   - **<u>Hint</u>**: Using the **Scanner** class involves 1) importing a package, 2) creating a Scanner object, and 3) using the appropriate method to read in a single line of text.
3. You will be splitting up (parsing) the information in the tweet into the 5 different types of information specified by the hashtags (type, location, detail, latitude, & longitude). Declare variables to store each of these pieces of information. Be sure to give these variables the appropriate data types and meaningful names.
4. Now you actually need to divide the information in the tweet into the separate substrings of information. Declare 2 variables (**start** and **finish**) to hold the indices of where each substring starts and finishes.
   - **<u>Hint</u>**: Each substring *starts* with a hashtag and *finishes* with a semicolon (Is there a String class method that you can use to get the index of "#typ" or a ';' ?)



5. Once you have numbers assigned to **start** and **finish**, we want to discard the #tag and extract only the value. We know that the ';' is where the value finishes – but we need to find the index where the actual *value* begins. Hint: our **start** variable currently points to the index of the "#" – and we know that all hashtag identifiers have the format, hashtag, 3 letters, and a space. Can we do simple math here to figure out the starting position of our *value?*
6. Once we have the correct starting and ending positions for the value, we want to extract the substring at those positions and assign it to the appropriate variable declared in step 3.
   - The **trim()** method removes leading and trailing white spaces (if any) from a **String**. Use the **t**rim() method on each resultant **String**.
   - **<u>Hint</u>**: The **trim()** method returns a modified **String**, but does not alter the **String** object that calls it. Ensure you are (re-)assigning a **String** to the result of **trim()**.

   Example

```
String original = "  text here   ";
String trimmed = original.trim(); //trimmed contains "text here"
//At this point, the original string still contains -- "  text here   "
```

7. After extracting the value encoded by each hashtag, we discard that part of the tweet String – we are finished with it and are ready to repeat these steps for the next hashtag. We can use the **substring** method to extract the substring of our tweet variable starting where the last hashtag *finished* **(** Hint: we know it finishes at a semicolon – and we have that index stored in our finish variable, and we want to start right **after** that semicolon – Also, remember that if we pass the substring method only 1 value, it begins at that index and goes until the end of the String).

DISCARD — SUBSTRING STORED IN TWEET

~~**#typ** *value* ;~~ | **#det** *value* ; **#loc** *value* ; **#lat** *value* ; **#lng** *value* ;

9. The **type** values come from a very small set of values, and we want to make them all caps. To do this, use the **toUpperCase** method of the **String** class.

10. We also want to ensure that the **detail** and **location** values are free of commas (which might pose an obstacle to further processing). Use the **replace** method of the **String** class to do this. Replace each comma with a single hyphen (-).

11. Now use **System.out** and **print** or **println** statements to produce formatted output, as shown in the included examples.
HINT: Use the escape sequence **\t** to include tabs as needed.

12. Your program's I/O must be the same as the I/O shown in the examples in the Example Program Input and Output section at the end of this document.

Additional Requirements

These are things that make the graders lives easier, and ultimately, you will see in the real world as well. Remember the teaching staff does not want to touch your code after they gave you requirements; they want to see the perfect results they asked for! Here is a checklist of things you can **lose points** for:

- (-1 point) If the source file(s)/class(es) are named incorrectly (case matters!)
- (-1 point) If your source file(s) have a package declaration at the top
- (-1 point) If any source file you submit is missing your Statement of Academic Honesty at the top of the source file. All submitted source code files must contain your Statement of Academic Honesty at the top of each file.
- (-1 point) If you have more than one instance of Scanner in your program. Your program should only have one instance of Scanner.
- (-1 point) Inconsistent I/O (input/output) that does not match our instructions or examples exactly (unless otherwise stated in an assignment's instructions). Your program's I/O (order, wording, formatting, etc.) must match our examples and instructions.
- (-2 points) If your submission is late
- (-7 points) If the source file(s) are not submitted before the specified deadline's late period ends (48 hours after the deadline) or if they do not compile.

- If your (-1 point) comments or (-1 point) variables are "lacking"
    - Here, "lacking" means that you or a TA can find **any** lines of code or variables that take more than 10 seconds to understand, and there is no comment, or the variable name does not make sense (variable names like **b**, **bb**, **bbb**, etc. **will almost never be acceptable)**
- (-1 point) Indentation is not consistent throughout your source code
    - Refresh your memory of indentation patterns in chapter 2 in the course textbook
    - Be careful of a combination of tabs and spaces in your files (use one or the other)!

If any of the above do not make sense to you, talk to a TA or your lab instructor.

eLC Submission and Grading

After you have completed and thoroughly tested **`ParseTheTweet.java`**, submit it to ***eLC*** in order to receive credit for the lab. Always double check that your submission was successful on ***eLC***!

The lab will be graded according to the following guidelines.

- Students may earn 3 points for lab attendance and 7 points if their program passes various test cases that we use for grading. **Some test cases will use values taken from the examples in this document and some test cases will not**. You should come up with additional test cases to check that your program is bug free.
- All source code must adhere to good Java programming style standards as discussed in lecture class and in the course textbook readings.
- Points will be deducted for not following any of the aforementioned instructions or requirements.

Example Tweets

Test your code with these. If you want to copy and paste the tweets below to your program, you'll need to move the input tweets to a text file first (a text file containing the input tweets are posted with this assignment), and copy one tweet at a time from text file to the input of your program. Do not copy and paste from a pdf file. <u>Copying and pasting from a pdf file may give runtime errors because of the location of the newline characters.</u>

1. `"#typ offer; #det free essential supplies 4 evacs pets.; #loc 2323 55th st, boulder; #lat 40.022; #lng -105.226;"`
2. `"#typ structure; #det damaged; #loc 224 left fork road (shed) (house okay); #lat 40.029854; #lng -105.391055;"`
3. `"#typ wind; #det just switched, and now the smoke is thick around our house; #loc 40.352,-105.2045; #lat 40.3523; #lng -105.2045;"`
4. `"#typ fire; #det firefighter sees a glow; #loc sw from west coach rd toward sunshine canyon; #lat 40.0515; #lng -105.332;"`
5. `"#typ structure; #det damaged; #loc unknown number, by 204 gold run road; #lat 40.050904; #lng -105.373941;"`
6. `"#typ evac; #det overflow shltr 4 evacuees at; #loc walt clark middle school, loveland; #lat 40.383; #lng -105.113;"`
7. `"#typ no fire; #det activity; #loc west of the pinewood res dam; #lat 40.367; #lng -105.292;"`

8. "#typ photo; #det local wild horse; #loc wild horse; #lat 40.052304; #lng -105.3193ch 74;"
9. "#typ smoke; #det its raining ash; #loc windsor, co; #lat 40.499812; #lng -105.012075;"

Example Program Input and Output

Each example below is one correct execution of a program. Your program's input and output must match what is below except the input tweet should be inputted on a single line in your program. In this document, each inputted tweet is word wrapped on multiple lines because it cannot be displayed on one line, but in your program, the input tweet should be inputted on a single line of text. Correct formatting of I/O is required for this assignment. As stated in the Example Tweets section, do not copy an input tweet from a pdf file and paste it into your program (copy each input tweet from a text file instead).

```
Enter a tweet below
"#typ offer; #det free essential supplies 4 evacs pets.; #loc
2323 55th st, boulder; #lat 40.022; #lng -105.226;"
Type:           OFFER
Detail:         free essential supplies 4 evacs pets.
Location:       2323 55th st- boulder
Latitude:       40.022
Longitude:      -105.226

Enter a tweet below
"#typ structure; #det damaged; #loc 224 left fork road (shed)
(house okay); #lat 40.029854; #lng -105.391055;"
Type:           STRUCTURE
Detail:         damaged
Location:       224 left fork road (shed) (house okay)
Latitude:       40.029854
Longitude:      -105.391055

Enter a tweet below
"#typ wind; #det just switched, and now the smoke is thick
around our house; #loc 40.352,-105.2045; #lat 40.3523; #lng -
105.2045;"
Type:           WIND
Detail:         just switched- and now the smoke is thick around
our house
Location:       40.352--105.2045
Latitude:       40.3523
Longitude:      -105.2045

Enter a tweet below
"#typ fire; #det firefighter sees a glow; #loc sw from west
coach rd toward sunshine canyon; #lat 40.0515; #lng -105.332;"
Type:           FIRE
Detail:         firefighter sees a glow
Location:       sw from west coach rd toward sunshine canyon
Latitude:       40.0515
```

```
Longitude:    -105.332

Enter a tweet below
"#typ structure; #det damaged; #loc unknown number, by 204 gold
run road; #lat 40.050904; #lng -105.373941;"
Type:         STRUCTURE
Detail:       damaged
Location:     unknown number- by 204 gold run road
Latitude:     40.050904
Longitude:    -105.373941

Enter a tweet below
"#typ evac; #det overflow shltr 4 evacuees at; #loc walt clark
middle school, loveland; #lat 40.383; #lng -105.113;"
Type:         EVAC
Detail:       overflow shltr 4 evacuees at
Location:     walt clark middle school- loveland
Latitude:     40.383
Longitude:    -105.113

Enter a tweet below
"#typ no fire; #det activity; #loc west of the pinewood res
dam; #lat 40.367; #lng -105.292;"
Type:         NO FIRE
Detail:       activity
Location:     west of the pinewood res dam
Latitude:     40.367
Longitude:    -105.292

Enter a tweet below
"#typ photo; #det local wild horse; #loc wild horse; #lat
40.052304; #lng -105.319374;"
Type:         PHOTO
Detail:       local wild horse
Location:     wild horse
Latitude:     40.052304
Longitude:    -105.319374

Enter a tweet below
"#typ smoke; #det its raining ash; #loc windsor, co; #lat
40.499812; #lng -105.012075;"
Type:         SMOKE
Detail:       its raining ash
Location:     windsor- co
Latitude:     40.499812
Longitude:    -105.012075
```