Lab Exercise #2
Regular Expressions and String.split()
Computer Science 2334

Group #: ____________ Section #: ____________

Members: ______________________________

____

____

____

____

Learning Objectives:

- To understand the use of regular expressions.
- To learn how to use *String.split()* and regular expressions to divide a string into multiple strings.
- To demonstrate this knowledge by completing a series of exercises.

Instructions:

This lab exercise requires a laptop with an internet connection. Once you have completed the exercises in this document, your group will submit it for grading. All group members should legibly write their names at the top of this lab handout. *Make sure you also sign the attendance sheet.*

The *String.split()* method in the *String* class will be extremely useful for completing the objectives in Project 1. The string processing capabilities of the *String* class will be extensively used in later projects as well.

Go to the Java API homepage at http://java.sun.com/j2se/1.5.0/docs/api/index.html. This page contains a listing of all the classes that make up the Java Application Programmer's Interface. The Java API can be used to find the answers to many of questions that will arise during the semester about how to design and implement programs. The API should be the first place you look for answers to questions.

Along the left hand side of the web page of the Java API is a listing of classes in alphabetical order. Find and click on the *String* class. The main window will show detailed information on the class. As you scroll down through this listing, you'll see summary information on the class, the various constructors for the class, and then the methods of the class. Look for the *split()* method that takes a single *String* as its only argument. Click on the method name to view detailed information about the *split()* method.

> *Tip:* You can search for a class in the listing on the left hand side in most browsers by clicking in the pane (frame) containing the listing and pressing the key combination CTRL+F to invoke the *"Find in Page"* feature.

String.split() has two forms, one that takes a single argument, and one that takes two arguments. Study the version that takes a single argument. This argument is a regular expression. Regular expressions are sets of symbols and syntactic elements that are used to match patterns of text. We're going to use some simple regular expressions (or *regex*es) to divide an input string into separate elements of a *String* array.

The *regex* in *split()* identifies the pattern of characters to use as separators in your *String* array. Let’s actually try this out.

In your group, write a small Java program that will consist only of a main method. This class must be named *SplitTest*. The program will have an input *String* variable, called *inputString* (you don’t need to use any particular input methods in this program), and a *String* array called *outputArray*. You should split the *inputString* into the *outputArray* with the following line of code:

```
        outputArray = inputString.split(“m”);
```

After this line of code, you should print the contents of *outputArray* to the console using a *for* loop.

What output do you get if the *inputString* is “`Premature optimization is the root of all evil in programming.`”?

What happens if the *inputString* is “`Premaature optimmmizzzation is the root of all evil in programming.`”?

Now change the call to *split()* from `split(“m”)` to `split(“[tz]”)`. What is the result now? What seems to be happening?

Square brackets form what is called a regular expression character class. When we use a *regex* like “`[tz]`” we are saying “either t or z”. A character class can be as simple as a list of different characters such as “`[ashf]`” which will match the letters a, s, h, or f. Much more complicated classes can be constructed, like ranges of characters “`[a-z]`” for lowercase ‘a’ to lowercase ‘z’.

Now run the *splitTest.java* program again with the following code changes:

```
        outputArray = inputString.split(“[m]+”);
```
How has the output changed? Why?

Take a look at the Regular Expression tutorial on Sun's web page found at http://java.sun.com/docs/books/tutorial/extra/regex/. The entire tutorial would be good to look over, but for now focus on the following sections:

- Introduction
- The Metacharacters subsection of String Literals
- Predefined Character Classes

What would be some useful Predefined Character Classes for splitting a string of text (that includes punctuation) into an array of words? What patterns do these classes match?

Design a *regex* that can match against all punctuation characters and whitespace characters and does not result in your array having an empty strings in it when you call *String.split(). You may find the tutorial and the Java API helpful for answering this question*. What is the *regex* that you have designed?

Test this expression with various types of input in order to see if it meets the objectives in the previous question.

What are some good input strings to test the *regex* in the previous question against?

Lab Submission:
To submit this project transfer your *SplitTest.java* file to codd.cs.ou.edu and save it in a folder named lab2. Then ssh into codd.cs.ou.edu and use the following commands to submit your lab.

```
    cd lab2
    /opt/cs2334/bin/submit cs2334-<section #> lab2 SplitTest.java
```

where <*section #*> is replaced by the section number for your lab section (011, 012, or 013).