CSCE 156 – Lab: Strings & File I/O

Java Handout

0. Prior to the Laboratory

1. Review the laboratory handout
2. Read Java String tutorial:
   http://download.oracle.com/javase/tutorial/java/data/strings.html
3. Read Java Arrays tutorial:
   http://download.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
4. Read Java File I/O tutorial:
   http://download.oracle.com/javase/tutorial/essential/io/file.html
5. Read the Wikipedia article on the general printf functionality and a tutorial on Java's implementation:
   http://en.wikipedia.org/wiki/Printf_format_string
   http://sharkysoft.com/archive/printf/docs/javadocs/lava/clib/stdio/doc-files/specification.htm

1. Lab Objectives

Following the lab, you should be able to:

- Use Strings and Arrays to write Java programs
- Use basic File I/O in Java

2. Lab Topics

- Strings, String manipulation, String operations & functions
- File input and output, formatted output

3. Problem Statement

You will familiarize yourself with Strings and File Input/Output by completing two Java programs.

The first program involves processing a DNA nucleotide sequence (a string consisting of the characters A, G, C, and T—standing for the nucleobases adenine, guanine, cytosine, and thymine). A common operation on DNA is searching for and counting the number of instances of a particular subsequence. For example, in the following DNA sequence,

T A G A A A A G G G A A A G A T A G T

the subsequence TAG appears twice. Your activity will involve processing a file containing a nucleotide sequence of the H1N1 flu virus and counting the number of instances of various subsequences.

The second program involves processing a file containing formatted data. Specifically, you will process a file containing the win/loss records of National League baseball Teams from the 2011 season. The file is formatted as follows: each line contains the win/loss record of a single team (16 teams total). Each line contains the team name, the number of wins, and the number of losses. Your program will read in the file, process the data and sort the teams in the order of their win percentage (wins divided by total games) and output the sorted and reformatted team list into a new file.

Formatted Output

Most programming languages support or implement the standard functionality of the "printf" standard of *formatted output* originally provided in the C programming language.

The printf function is a variable argument function that takes a string as its first argument that specifies a format in which to print the subsequent arguments. Various flags can be used to print variable values in a specific format or as a specific type. Some of the major flags supported:

- %Ns – print the argument as a string, right-justified with at least N characters. If the string value is less than N, it will be padded out with spaces. Variations on this flag can be used to change the padding character and to left-justify instead.
- %Nd – print the argument as an integer with at least N spaces.
- %N.Mf – print the argument as a floating point number with at least N characters (including the decimal) and at most M decimals of precision.

Consider the following example code snippet.

```
String a = "hello";
int b = 42;
double c = 3.1418;
String result = String.format("%10s, %5d\t%5.2f\n", a, b, c);
System.out.println(result);
```

The resulting line would be (note the spacing):

```
     hello,    42       3.14
```

Activity 1: Substring Searching

Instructions

1. Download the `DNA.java` and H1N1nucleotide.txt files from Blackboard and import them into an Eclipse project. To import the text file do the following:
   - Right-click your project and create a new folder called "data"
   - Drag and drop the text file into this folder
2. Modify the code to read in the subsequence from the command line (and to echo an error and exit if it is not provided).
3. The code to read in and process the nucleotide sequence is already provided. Study its usage: it reads in the file line by line, trims it (removes leading and trailing whitespace) and concatenates

it into one large string. Note that the `Scanner` class is used in conjunction with the `File` class.

4. At the end of the program, write your own subsequence counting code by writing two for-loops to count the number of instances of `subsequence`. Note: individual characters of Strings in Java can be obtained by using the `charAt` method; `charAt(0)` for example, gives you the first character in the string. Individual characters can be compared using the == operator.
5. Alternatively, you may leverage the methods and functions that the Java `String` class provides to do some of the grunt work for you.
6. Demonstrate your working code to a lab instructor and answer the question(s) in your worksheet.

Activity 2: Baseball Records

Instructions

1. Download the `Team.java` and `Baseball.java` files from Blackboard and include them in your Eclipse project.
2. Much of the code has been provided for you, including code to print the teams for debugging purposes and to sort the teams according to win percentage.
3. Write code to read in the mlb_nl_2011.txt data file at the appropriate point in this `Baseball.java` program. Note:
   - You can use the Scanner class to get individual tokens. By default the delimiter for this class is any whitespace. For example, you may find the `hasNext()`, `next()` and `nextInt()` methods useful.
   - The Team class has one constructor that takes the team name, wins, and losses; example:
     ```
      Team t = new Team(name, wins, loss);
     ```
   - If you have a team instance, you can make use of its methods using the following syntax:
     ```
     String name = t.getName();
     double winPerc = t.getWinPercentage();
     ```
4. Write code to output the sorted array of teams to a file called “mlb_nl_2011_results.txt” in the data folder according to the following format:
   - Each team’s information should appear on a separate line.
   - Each line should contain the team’s name and its win percentage.
   - The team name should be right-justified in a column of length 10, the win percentage should be a number 0 – 100 with two decimals of precision.
   - Note: it may be necessary to refresh your project after running your program (select the data folder and press F5).
5. Hint: to output to a file, use the `PrintWriter` class which supports easy output to files. A full example:
   ```
   try {
     PrintWriter out = new PrintWriter(“filename”);
     out.write(“you can output a string directly using this
   method!”);
   ```

```
        out.close();
      } catch (FileNotFoundException fnfe) {
        fnfe.printStackTrace();
      }
```

6. Your output file should look like the following:

```
...
 Cardinals 55.56
    Braves 54.94
    Giants 53.09
   Dodgers 50.93
 Nationals 49.69
...
```

7. Answer the questions in your worksheet and demonstrate your working code to a lab instructor.

Advanced Activity (Optional)

The code to sort the teams according to their win percentage was provided for you. It involved defining a `Comparator` (as an anonymous class) that was passed as an argument to a built-in sort method. Study this code and read the documentation for the sorting function. Modify the comparison function to sort the list of teams in alphabetic order according to the team name.