CSCE 156 – Computer Science II

Lab 3.0 - Strings & File I/O

Prior to Lab

Review this laboratory handout prior to lab.

For Java:

1. Read Java String tutorial:
   http://download.oracle.com/javase/tutorial/java/data/strings.html

2. Read Java Arrays tutorial:
   http://download.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

3. Read Java File I/O tutorial:
   http://download.oracle.com/javase/tutorial/essential/io/file.html

4. Read the Java tutorial on the `System.printf()` method:
   https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

For PHP:

1. Read PHP Arrays tutorial:
   http://www.php.net/manual/en/language.types.array.php

2. Read PHP tutorial on string functions:
   http://www.php.net/manual/en/ref.strings.php

3. Read PHP tutorial on file streams:
   http://www.php.net/manual/en/book.filesystem.php

4. Read the PHP Manual on the `printf()` (print formatted) and `sprintf()` functions:
   http://php.net/manual/en/function.printf.php
   http://www.php.net/manual/en/function.sprintf.php

Lab Objectives & Topics

Following the lab, you should be able to:

- Use Strings and Arrays in PHP/Java
- Use basic File I/O in PHP/Java

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

Getting Started

Clone the project code for this lab from GitHub in Eclipse using the URL, https://github.com/cbourke/CSCE156-Lab03. Refer to Lab 1.0 for instructions on how to clone a project from GitHub.

For those with prior Java experience, do the PHP section. For those without prior Java experience, do the Java section.

PHP

Strings & File IO

You will familiarize yourself with strings and file input/output by completing two PHP scripts.

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,

$$TAGAAAAGGGAAAGATAGT$$

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 (negative N) 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.

```php
1   $a = "hello";
2   $b = 42;
3   $c = 3.1418;
4   printf("%10s, %5d\t%5.2f\n", $a, $b, $c);
```

This would result in the following (note the spacing denoted with a ␣ symbol):

```
>␣␣␣␣␣hello,␣␣␣␣␣42          ␣3.14
```

Activities

Substring Searching

1. Open the `PHP/dna.php` and the `data/H1N1nucleotide.txt` files

2. Modify the code to read in a 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 each line (removes leading and trailing whitespace) and concatenates it into one large string.

4. At the end of the script, write your own code to count the number of occurrences of the subsequences. Strings in PHP can be processed like character arrays: `$subsequence[0]` gives you the first character of `$subsequence`. Individual characters can be compared using the `==` comparison operator.

5. Hand in your source file using the webhandin and grade yourself with the webgrader.

Baseball Records

1. Open the `PHP/Team.php` and `PHP/baseball.php` files. The `Team` class has already been implemented and included in the `baseball.php` script.

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 `data/mlb_nl_2011.txt` data file at the appropriate point in this script. Each line in this file contains a team name, number of wins, and number of losses. You may find the following functions useful:

- `preg_split($pattern, $str)` (http://php.net/manual/en/function.preg-split.php)

- `explode($delimiter, $str)` (http://www.php.net/manual/en/function.explode.php)

Both functions tokenize the given string according to a regular expression or a delimiter respectively and return an array of string tokens.

4. The `Team` class has one constructor that takes the team name, wins, and losses; example:

```
$team = new Team($name, $wins, $loss);
```

You can access the methods of the `Team` class using the syntax:

```
1  $name = $team->getName();
2  $winPerc = $team->getWinPercentage();
```

5. Write code to output the sorted array of teams to a file `data/mlb_nl_2011_results.txt` 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.

Your output file should look like the following:

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

6. Answer the questions in your worksheet and demonstrate your working programs 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* function that was passed as an argument to a built-in sort function. 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.

Java

Strings & File IO

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,

$$TAGAAAAGGGAAAGATAGT$$

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 (negative N) 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.

```java
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);

//alternatively:
System.out.printf("%10s, %5d\t%5.2f\n", a, b, c);
```

This would result in the following (note the spacing denoted with a ␣ symbol):

```
>␣␣␣␣␣hello,␣␣␣␣␣42          ␣3.14
```

Activities

Substring Searching

1. Open the `DNA.java` and the `data/H1N1nucleotide.txt` files

2. Modify the code to read in a 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 each line (removes leading and trailing whitespace) and concatenates it into one large string.

4. At the end of the script, write your own code to count the number of occurrences of the subsequences. Individual characters in a Java `String` can be retrieved using the `charAt(int)` method. For example, `subsequence.charAt(0)` gives you the first character of the string `subsequence`. Individual characters can be compared using the `==` comparison operator.

5. Hand in your source file using the webhandin and grade yourself with the web-grader.

Baseball Records

1. Open the `Team.java` and `Baseball.java` files. The `Team` class has already been implemented for you.

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 `data/mlb_nl_2011.txt` data file at the appropriate point in this script. Each line in this file contains a team name, number of wins, and number of losses. You may find the following useful:

   - The `Scanner` class can be used to get individual tokens in a file. By default the delimiter for this class is any whitespace.

   - 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 an instance of `Team`, you can make use of its methods using the following syntax:

     ```
     1  String name = t.getName();
     2  double winPerc = t.getWinPercentage();
     ```

4. Write code to output the sorted array of teams to a file `data/mlb_nl_2011_results.txt` 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.

   Your output file should look like the following:

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

   Hint: to output to a file, use the `PrintWriter` class which supports easy output to files. A full example:

```
1  PrintWriter out;
2  try {
3    out = new PrintWriter("filename");
4  } catch (FileNotFoundException fnfe) {
5    throw new RuntimeException(fnfe);
6  }
7  out.write("you can output a string directly using this method!");
8  out.close();
```

5. Answer the questions in your worksheet and demonstrate your working programs 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 method. Modify the `Comparator` to sort the list of teams in alphabetic order according to the team name.