CSCE 156 – Lab03: Strings, File I/O & Conditionals

Java Handout

0. Prior to the Laboratory

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 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
5. Read for loop tutorial:
   http://download.oracle.com/javase/tutorial/java/nutsandbolts/for.html
6. Read while/do while loop tutorial:
   http://download.oracle.com/javase/tutorial/java/nutsandbolts/while.html

1. Lab Objectives

Following the lab, you should be able to:

- Use Strings and Arrays to write Java programs.
- Use basic File I/O operations and generate formatted output in Java.

2. Problem Statement

You will familiarize yourself with Strings and File Input/Output by completing the following Java program.

The 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: Baseball Records

Instructions

1. Download the `Team.java` and `Baseball.java` files from Canvas and include them in your Eclipse project. Also download the mlb_nl_2011.txt and mlb_nl_2011_results.txt files from Canvas and import them as follows:
    a. Right-click your project and create a new folder called "data"
    b. Drag and drop the text files into this folder
1. 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.
2. 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();
```

3. 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).
4. 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();
}
```

5. 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 code to a lab instructor.

3. Problem Statement

Java provides standard control structures for conditionals and repetition. Specifically, Java provides the usual if-then-else statements and while, for, and do-while loops. The syntax for these control structures should look familiar; some examples:

```
if(condition1) {
  //DO SOMETHING
} else if(condition2) {
  //DO SOMETHING ELSE
} else {
```

```
  //OTHERWISE
}

for(int i=0; i<n; i++) {
  //DO SOMETHING
}

int i=0;
while(i<n) {
  //DO SOMETHING
  i++;
}

int i=0;
do{
  //DO SOMETHING
  i++;
} while(i<n);
```

In addition, Java provides a foreach-loop, also referred to as an *enhanced for-loop*, for iterating over collections (classes that implement the `Iterable` interface) or elements in an array. This feature is mostly for convenience. The following examples illustrate this loop's usage.

```
String arr[] = new String[10];
...
for(String s : arr) {
  System.out.println(s);
}
```

Activity 2: Child Tax Credit

When filing for federal taxes, a credit is given to tax payers with dependent children according to the following rules. The first dependent child under 18 is worth a $1000.00 credit. Each dependent child younger than 18 after the first is worth a $500 tax credit each. You will complete a Java program to output a table of dependent children, how much each contributes to a tax credit, and a total child tax credit. Your table should look *something* like the following.

```
Child            Amount
Tommy (14)       $1000.00
Richard (12)     $500.00
Harold (21)      $0.00
Total Credit:    $1500.00
```

Instructions

1. Download the `Child.java` and `ChildCredit.java` files from Canvas to a project in Eclipse.

2. The Child class has already been implemented for you. Note how the Child class is used in the `ChildCredit` main method; several instances of children have been created and placed into an array.
3. Write code to iterate over the array, compute the child tax credits and output a table similar to the one above. Note: to call a method on an instance of the Child class, use the following syntax: `kid.getAge()`
4. Answer the questions in your worksheet and demonstrate your working code to a lab instructor.

Advanced Activity (Optional)

Use the `String.format` method to reformat the output of the Child Tax Credit program to print every piece of data in its own column.