CSCE 156 – Lab02 : Conditionals, Repetition & String

Java Version

0. Prior to the Laboratory

1. Read the following Java tutorials:
   - *If-then-else*: http://download.oracle.com/javase/tutorial/java/nutsandbolts/if.html
   - *Switch-case:* http://download.oracle.com/javase/tutorial/java/nutsandbolts/switch.html
   - *For loop:* http://download.oracle.com/javase/tutorial/java/nutsandbolts/for.html
   - *While/Do-while loop:* http://download.oracle.com/javase/tutorial/java/nutsandbolts/while.html
   - *Strings:* http://download.oracle.com/javase/tutorial/java/data/strings.html
   - *Arrays:* http://download.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

1. Lab Objectives

Following the lab, you should be able to:

- Use if-then-else statements to control the logical flow of the program.
- Use switch-case statement to control the logical flow of the program.
- Use for/while loops to implement repetition statements in your program.
- Use Strings and Arrays to write Java programs.
- Write complex Java programs that require conditional logical statements and or loops.
- Import and use a Java library within the IDE.

2. 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 1: Sum of Natural Numbers

Natural numbers are the usual counting numbers; 1, 2, 3, … In this exercise you will write several loops to compute the sum of natural numbers 1 thru n where n is read from the command line. You will also write a *foreach* loop to iterate over an array and process data.

Instructions

1. Download the `Natural.java` file from Canvas and import it into a project in Eclipse.
2. You'll note that code to read in *n* has already been provided for you. An array mapping integer values 0 thru 10 to text values has also been created for you.
3. Write a for-loop and a while-loop to compute the sum of natural numbers 1 thru n and output the answer.
4. Write an enhanced for-loop loop to iterate over the elements of the `zeroToTen` array. As you iterate over the elements, concatenate each string, delimited by a single space to a result string and print the result at the end of the loop. Your result should look something like the following:
   `zero one two three four five six seven eight nine ten`
5. Demonstrate your working code to a lab instructor and have them sign your worksheet.

3. Problem Statement

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

The 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.

Activity 2: Substring Searching

Instructions

1. Download the `DNA.java` and H1N1nucleotide.txt files from Canvas and import them into an Eclipse project. To import the text file do the following:
    a. Right-click your project and create a new folder called "data"
    b. 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 for-loop 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. Hand in your source file using the webhandin and grade yourself with the webgrader.

Activity 3: Birthday Program & Libraries

No man is an island. Good code depends on selecting and (re)using standard libraries whenever possible so that you are not continually reinventing the wheel. This activity will familiarize you with how to import and use a Java library. Java libraries are usually packaged into JAR (Java ARchive) files.

Instructions

1. Import (drag and drop) the Birthday.java file from Canvas as described previously.
2. You will notice there are compiler errors because this class uses other classes that are not available in the standard JDK (Java Developer Kit). It instead uses classes from the Joda-Time

library; a library of useful classes and utilities for dealing with dates, times, intervals, durations and more.

3. Download the `joda-time-2.0.jar` file from Canvas (save to your desktop).
4. Create a library folder in your project:
    - Right-click the project and select New -> Folder
    - Name the folder "lib" (short for library)
5. Import the Joda-Time library by dragging and dropping the jar file into this new lib folder.
6. Right-click the new JAR file and select Build Path -> Add to Build Path
7. The library is now imported and the compiler errors should go away.

Finishing the program

Though the program should have no syntax errors, if you run it, no output will be displayed. You need to complete the program as follows.

1. For the variables, `name`, `month`, `date`, and `year`, enter your own information (your name and your birthday).
2. Add appropriate code (using `System.out.println` which prints to the standard output a full line) a greeting like:
   Greetings, NAME. Today you are XX years, XX months, and XX days old.
   Of course, XX should be replaced with variable values. Note: In Java, variable values can be concatenated with strings using the + (plus) operator.
3. Add a conditional statement that, if today is the user's birthday will output "Happy Birthday". If it is not the user's birthday, output "Your friends have XX shopping days until your next birthday" where XX is replaced with the appropriate variable value.

Complete your worksheet and demonstrate your working code to a lab instructor.

Advanced Activity (Optional)

Explore the Joda-Time library (its API is available at: http://joda-time.sourceforge.net/api-release/index.html) and Java itself by implementing the following program. The program will read in (or hard code) two birthdays and compute the number of years, months, and days between them.