CSCI165 Computer Science II
Homework Assignment: Java Primitives
Due: Monday 2/3/2020 by 9:00 AM

Save all source code files into the "hw" directory inside "week-2" of your local repository

Complete the following problems 1 - 5 in a file called: **Primitives.java**

1. Define and initialize variables of each of the Java primitive types. Use appropriate sample data that is not the default values. Print each value with a descriptive method using printf. Demonstrate both character and numeric literals for the **char** type. Demonstrate some widening and narrowing type casts.
2. Ask the user to enter an integer and display the square, cube, and fourth power. Research the **Math** class and use the **pow** method for each calculation. Use a loop if you'd like.
3. Create two variables of type **int**. Assign these variables the maximum and minimum values of this data type. Use the **MIN_VALUE** and **MAX_VALUE** defined constants in the Integer class. Research the API for this.
    1. Print the values
    2. Research and experiment with the **compare** and **compareUnsigned** methods of the Integer class. Demonstrate that you can call these methods correctly. Display the results along with a descriptive message. Include comments that describe the issues surrounding comparing signed and unsigned values.
4. Write a Java code to test whether a given double/float value is a finite floating-point value or not. Research the **isFinite** method of the ***Float*** and ***Double*** classes. Each primitive type has an associated class that contains methods that operate of values of that type. These classes are called wrappers, as they wrap a primitive value with appropriate methods. A floating point number is finite if it has a fixed number of fractional digits ie. a rational number. Ask the user to enter two floating point numbers. Store one in a variable of type **double**, store the other one in a variable of type **float**. Define the fraction ***0/number*** and pass the result into the appropriate **isFinite** method. Store the result of that method call into a variable of type **boolean**. Print the result with a descriptive message using **printf.** Demonstrate a value that is finite and one that is not.
5. Ask the user to enter a integer dividend and divisor. Compute floor division and the floor modulus. Use both the operators (/ and %) and the **floor** methods from the Math class. Look this up in the API. Print the result with a descriptive message using **printf**
6. Create a ***new source file*** called **GMT.java**. Write Java code that prints the current time in GMT. GMT: Greenwich Mean Time (GMT) is the mean solar time at the Royal Observatory in Greenwich, London. GMT was formerly used as the international civil time standard, now superseded in that function by Coordinated Universal Time (UTC). Every inhabited place in the world has a UTC offset that is a multiple of 15 minutes, and the majority of offsets (as well as all nautical time zones) are measured in whole hours. There are many cases where the national

civil time (ignoring Daylight Saving) uses a UTC offset (time zone) that is different from the theoretical one appropriate to its longitude.

1. **Time Zone Offsets:** https://upload.wikimedia.org/wikipedia/commons/e/e8/Timezones2008_UTC%2B2_gray.png
2. Use the **currentTimeMillis** method from the System class. Research the API.
3. Ask for the time zone offset to GMT (-5 would be New York). Inspect the TimeZone Offset map linked above to get a grasp of this concept.
4. You are not allowed to use if statements. Construct an appropriate mathematical expression.
5. Sample Run at 6:37 PM:
   Input the time zone offset to GMT: -5
   Current time is 18:37:59

7. Create a new source file called **Initials.java**. Write Java code that asks the user for their first and last name. Store these in a **single variable.** Extract the first character of the first name into a variable of type **char**. Extract the first character of the last name into a variable of type **char.** Use the String method **indexOf** to locate the space. Print
   1. The characters individually
   2. The numeric values of the characters (Unicode value)
   3. The sum of the numeric values
   4. The characters concatenated together as a String.

Submission:

Push to your remote repo.