Lab Partner 1 Name
Lab Partner 2 Name

CS 115 Fall 2019 Lab #3

Due: **Monday, September 16th, 5:00 PM**
Points: **20**

Instructions:

1. Use this document template to report your answers. Enter all lab partner names at the top of first page.
2. You don't need to finish your lab work during the corresponding lab session.
3. ZIP your lab report and java files (if any) into a single ZIP file. Name the ZIP file as follows:

```
LastName_FirstName_CS115_Lab3_Report.zip
```

4. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.
5. ALL lab partners need to submit a report, even if it is the same document.

Objectives:

1. (0 points) Enter, compile, and run your first java program using Notepad++.
2. (2 points) Learn to recognize syntax error messages.
3. (3 points) Demonstrate the ability to declare and initialize variables.
4. (15 points) Use arithmetic operators. Understand and apply precedence of operations. Understand and apply explicit casting.

Problem 1 [0 points]:

Enter, compile, and run your first java program using Notepad++.

First, open Notepad++ and type in or copy the below program code into a blank document:

```
public class Hello {
  public static void main (String[] args) {
    System.out.println("Hello World!");
  }
}
```

Next, save your file as

```
Hello.java
```

and note that class name matches the file name.

Note: Any files that you create / save on lab computers will be deleted once you log off. You should either use a thumb drive or upload to your google drive / email it to yourself to retain the files.

Now:

- From the Notepad++ menu items at the top choose "Macro -> Compile Java". In the Console window at the bottom you will see the "javac Hello.java" command executed along with any compiler messages including **any compile errors**. If everything is OK, the message "Process finished. (Exit code 0)" should appear. Note: whenever you compile, Notepad++ will automatically save the file first.
- From the menu items at the top choose "Macro -> Run Java". In the Console window at the bottom you will see the "java Hello" command executed including **any runtime errors**. If everything is OK, the output "Hello World!" and the message "Process finished. (Exit code 0)" should appear. Congratulations, you have sucessfully compiled and run your first java program.

Problem 2 [2 points]:

Learn to recognize syntax error messages.

First, open Notepad++ and type in or copy the below program code into a blank document:

```
public class TryVariables {

     // This example shows how to declare and initialize variables
     public static void main( String [] args ) {
          int testGrade = 100;
          long cityPopulation = 425612340;
          byte ageInYears = 19;
          float  salesTax = .05F;        // the trailing F makes a float numeric literal
          double interestRate = 0.725
          double avogadroNumber = 602214076000000000000000.;
          char finalGrade = 'A';
          boolean isEmpty = true;

          System.out.println( "testGrade is " + testGrade );
          System.out.println( "cityPopulation is " + cityPopulation );
          System.out.println( "ageInYears is " + ageInYears );
          System.out.println( "salesTax is " + salesTax );
          System.out,println( "interestRate is " + interestRate );
          System.out.println( "avogadroNumber is " + avogadroNumber );
          System.out.println( "finalGrade is " + finalGrade );
          System.out.println( "isEmpty is " + isEmpty );
     }
}
```

Next, save your file as

`TryVariables.java`

and note that class name matches the file name.

This code has two syntax errors introduced on purpose. Your task is to find and fix them:

- Try to locate syntax errors and correct them,
- From the Notepad++ menu items at the top choose "Macro -> Compile Java". In the Console window at the bottom you will see the "javac TryVariables.java" command executed along with any compiler messages including **any compile errors**. If everything is OK, the message "Process finished. (Exit code 0)" should appear. Note: whenever you compile, Notepad++ will automatically save the file first.
- From the menu items at the top choose "Macro -> Run Java". In the Console window at the bottom you will see the "java TryVariables" command executed including **any runtime errors**. If everything is OK, you should this on your screen:

Sample output:

```
testGrade is 100
cityPopulation is 425612340
ageInYears is 19
salesTax is 0.05
interestRate is 0.725
avogadroNumber is 6.02214076E23
finalGrade is A
isEmpty is true
```

Note that avogadroNumber value is formatted using scientific notation since it is such a large number.

Solution:

```
public class TryVariables {

    // This example shows how to declare and initialize variables
    public static void main( String [] args ) {
        int testGrade = 100;
        long cityPopulation = 425612340;
        byte ageInYears = 19;
        float  salesTax = .05F;        // the trailing F makes a float numeric literal
        double interestRate = 0.725;   // missing semicolon
        double avogadroNumber = 602214076000000000000000.;
        char finalGrade = 'A';
        boolean isEmpty = true;

        System.out.println( "testGrade is " + testGrade );
        System.out.println( "cityPopulation is " + cityPopulation );
        System.out.println( "ageInYears is " + ageInYears );
        System.out.println( "salesTax is " + salesTax );
        System.out.println( "interestRate is " + interestRate ); // , instead of .
        System.out.println( "avogadroNumber is " + avogadroNumber );
        System.out.println( "finalGrade is " + finalGrade );
        System.out.println( "isEmpty is " + isEmpty );
    }
}
```

Problem 3 [3 points]:

Create, compile, and run your first java program that declares and initializes variables (assigns intial values to it) using Notepad++.

First, open Notepad++ and type in or copy the below program code into a blank document:

```
public class Declaration {
  public static void main (String[] args) {

    // Enter your variable declaration statements below that line


    // Enter your variable initialization statements below that line



    // Display it
    System.out.println("IIT Zip Code is: " + IITZipCode);
    System.out.println("Sample UK postal code is: " + sampleUKPostalCode);
    System.out.println("My age is: " + myAge);
    System.out.println("Desired CS116 grade: " + myGrade);
    System.out.println("Earth-Moon distance in yards: " + earthMoonDistanceInYards);

  }
}
```

Next, save your file as

`Declaration.java`

and note that class name matches the file name.

Now:

- Below the comment line that says:

```
// Enter your variable declaration statements below that line
```

  add variable declaration statements (`dataType variableName;`) for each of the data types you selected in Lab #1 Problem 3 report. One line - one declaration statement. **Don't forget to add semicolon at the end of each statement! Note that variable names are used in following statements. Use those.**
- Below the comment line that says:

```
// Enter your variable initialization statements below that line
```

  add variable initialization statements (`variableName = value;`) for each of the data variables. One line - one initialization statement. Use 60616 as IIT ZIP code W2 2SZ as UK sample postal code. **Don't forget to add semicolon at the end of each statement!**
- From the Notepad++ menu items at the top choose "Macro -> Compile Java". In the Console window at the bottom you will see the "javac Declaration.java" command executed along with any compiler messages including **any compile errors**. If everything is OK, the message "Process finished. (Exit code 0)" should

appear. Note: whenever you compile, Notepad++ will automatically save the file first.

- Correct your data types if necessary (if you receive any compiler error messages related to it),
- From the menu items at the top choose "Macro -> Run Java". In the Console window at the bottom you will see the "java Declaration" command executed including **any runtime errors**. If everything is OK, you should see several sentences stating current values of your variables on screen:

Sample output:

```
IIT Zip Code is: 60616
Sample UK postal code is: W2 2SZ
My age is: 20
Desired CS116 grade: A
Earth-Moon distance in yards: 4.20388232721E8
```

Sample solution:

```
public class Declaration {
  public static void main (String[] args) {

     // Enter your variable declaration statements below that line
     int IITZipCode;
     String sampleUKPostalCode;
     byte myAge;
     char myGrade;
     double earthMoonDistanceInYards;

     // Enter your variable initialization statements below that line
     IITZipCode = 60616;
     sampleUKPostalCode = "W2 2SZ";
     myAge = 20;
     myGrade = 'A';
     earthMoonDistanceInYards = 420388232.721;

     // Display it
     System.out.println("IIT Zip Code is: " + IITZipCode);
     System.out.println("Sample UK postal code is: " + sampleUKPostalCode);
     System.out.println("My age is: " + myAge);
     System.out.println("Desired CS116 grade: " + myGrade);
     System.out.println("Earth-Moon distance in yards: " + earthMoonDistanceInYards);

  }
}
```

Problem 4 [15 points]:

GRADING: Please note the following if the submission is missing it or using incorrectly when you are grading and giving feedback - comments, descriptive identifier names, correct data types, unit coversion, String literals for output using System.out.prinntln(), constants, order of operations, mixed type arithmetic (implicit type casting), % operator (mod or remainder), explicit type casting.

Use arithmetic operators. Understand and apply precedence of operations. Understand and apply explicit casting.

For each of the following three problems complete:

1. Input-Process-Output Design (answer the questions below, fill the inputs and outputs table, and come up with pseudocode) **[1 out of 5 points each]**,
2. Test Plan Table (fill the Test plan table as for the previous lab) **[1 out of 5 points each],**
3. Java Program: Implement your Input-Process-Output Design in java. Make sure to utilize these programming concepts correctly when appropriate (comments, descriptive identifier names, correct data types, unit coversion, String literals for output using System.out.println(), constants, order of operations, mixed type arithmetic (implicit type casting), % operator (mod or remainder), explicit type casting). **Since we have not covered user input yet, your program should just assign values to input variables**. Compile, correct syntax errors, and run your program on all the test cases in your test table. Your results may differ slightly in the number of decimal digits displayed. Later in the term you will learn how to format output to a specified number of digits. **[3 out of 5 points each]**.

Note: you can reuse some of your answers, including test cases from previous lab if applicable.

INPUTS: What are the inputs?

- What format / data type are they? (integer, real number, single character, string - a sequence of characters)
- Any valid / invalid / illegal / special values? (positive, negative, valid range, etc.)
- How do you get them? (enter manually, ask user, read from file, etc.)

PROCESS: How do you get from inputs to the outputs you want?

- What are the calculation steps?
- To follow these steps, what else do you need? (formulas, etc.)
- Other variables, constants, conversions (besides input and output variables)

OUTPUTS: What are the outputs?

- What format / data type are they in? (integer, floating-point, character, or string)
- Any valid / invalid / illegal / special values? (positive, negative, valid range, etc.)
- How do you output them? (display on screen, save to a file, plot, tabularize, etc.)

A. The bank wants to be sure you can afford to pay them back before they give you a mortgage. One way they consider your ability to repay is by making sure your total debt doesn't exceed a certain percentage of your income, usually 36-42%. This percentage is called the debt ratio. Given your income (a real number) and other monthly debt (a real number), you can use the following formulas to determine the lower and upper limits on your monthly mortgage payments. **[5 points]**

- Lower limit = (36% of your income) minus your other monthly debt
- Upper limit = (42% of your income) minus your other monthly debt

Inputs and outputs (use “N/A”, “undefined”, “none”, etc. If necessary)					
Variable name	Input or Output ?	Data type / format	Constraints	Special cases	Comments

Pseudocode:

```
INPUT – monthlyIncome real>0          monthlyDebt    real>0
PROCESS    need to also know the 36% and 42% CONSTANTS
Lower limit = (36% of your income) minus your other monthly debt
Upper limit = (42% of your income) minus your other monthly debt
OUTPUT
Lower limit      real>0, 2 decimal places
Upper limit    real>0, 2 decimal places

Test Case Reason       Sample Data
High Debt           income: 4000
                               monthly debt:1000

Low Debt            income: 6000
                           monthly debt: 500

No Debt                    income: 3000
                           monthly debt: 0
```

Test case name (ex. “negative height”, “typical conditions”, etc.)	**Input data set for this test case**	**Explain why you chose this test case**

Sample solution:

```
public class DebtRatio {
      public static void main(String[] args) {
            double income=4000, otherMonthlyDebt=1000;
            double lowLimit, highLimit;
            final double LOW_PCT=.36, HIGH_PCT=.42;    //use of constants necessary

            lowLimit = (LOW_PCT*income) - otherMonthlyDebt;
            highLimit = (HIGH_PCT*income) - otherMonthlyDebt;

            System.out.println( "Lower Limit: $" + lowLimit);
            System.out.println( "High Limit: $" + highLimit);
      }
}
```

B. Some private water wells produce only 1/3 of a gallon of water per minute. Each member of a family will use around 60 gallons of water per day. One way to avoid running out of water with these low-yield wells is to use a separate, above-ground holding tank. However, there is also a "natural" water holding tank in the casing (i.e. the cylindrical hole) of the well itself. The deeper the well, the more water that will be stored in the casing that can be pumped out for household use. But water will not fill the entire depth of the well, in practice the static water level will generally be 50 feet below the ground surface.
We want to calculate the "tankSize" which is the minimum size of the separate, above-ground holding tank so there will be enough water in the well and the above-ground holding tank. at the start of the day for one day's use. The three inputs are:

- The real number radius of the well casing in inches (usually 2-6 inches).
- The real number total depth of the well in feet (usually 50-250 feet).
- The number of family members. **[5 points]**

Inputs and outputs (use "N/A", "undefined", "none", etc. If necessary)					
Variable name	Input or Output ?	Data type / format	Constraints	Special cases	Comments

Pseudocode:

```
input - radius (real 2-6 inches), depth (real 50-250), familySize (integer>0)
constants - 60 gallons/person/day, pi=3.14159, 12 inches/foot, 7.48 gallons/cubic foot
process -
  Volume of a cylinder = pi * height * radius^2
  calculate volume of the well = pi*(depth-50)*(radius/12)*(radius/12)
  calculate gallons of the well = volumeWell*7.48
  calcualte gallons needed by the family = 60*familySize
  calculate gallons neede for above ground tank = (gallons needed by the fmaily)-(gallons of the well)
output - gallons neede for above ground tank, real number, should be positive unless an above ground tank is not needed

Test data should include at least 3 cases
where the function returns a positive number (tank is needed)
where the function returns a negative number (NO tank is needed)
where the function returns zero, or close to it    (NO tank is needed)
```

Test case name (ex. "negative height", "typical conditions", etc.)	**Input data set for this test case**	**Explain why you chose this test case**

Sample solution:

```
public class TankSize {
	public static void main(String[] args) {
		double radius=3.5, depth=125;
		int familySize=5;
		double volumeWell, gallonsWell, gallonsNeeded;
		int gallonsFamily;
		final double MY_PI=3.14159, GAL_PER_CUBIC_FOOT=7.48;	//use of constants necessary
		final int GAL_PER_PERSON=60, INCHES_PER_FOOT=12, WATER_LEVEL=50;

		volumeWell = MY_PI*(depth-WATER_LEVEL)*(radius/INCHES_PER_FOOT)*(radius/INCHES_PER_FOOT);
		gallonsWell = volumeWell*GAL_PER_CUBIC_FOOT;
		gallonsFamily=GAL_PER_PERSON*familySize;
		gallonsNeeded=gallonsFamily-gallonsWell;

		System.out.println( "Tank Size Needed = " + gallonsNeeded + " gal.");
	}
}
```

if tankSize of zero or negative is calculated, then number no separate, above-ground holding tank is needed.

C. Most stop watches allow you to display the time elapsed as a number of seconds, or as hours:minutes: seconds. So 5437 seconds or 1 hour:30 minutes:37 seconds Given an integer number of seconds, calculate the equivalent integer number of hours, integer number of minutes, and integer number of seconds. **[5 points]**

Inputs and outputs (use "N/A", "undefined", "none", etc. If necessary)					
Variable name	Input or Output ?	Data type / format	Constraints	Special cases	Comments

Pseudocode:

```
1.What are the inputs?    seconds
  What format are they in?    positive integer
2.How do you get from inputs to the outputs you want (process)?
  What are the calculation steps?
  hours = floor(seconds/3600)
  leftover = mod(seconds, 3600)    or    leftover=leftover-(hours*3600)
  minutes = floor(leftover/60)
  leftover = mod(leftover,60)    or    leftover=leftover-(minutes*60)
  To follow these steps, what else do you need?
      Other variables, constants, conversion? (besides input and output variables)
      The number of seconds in an hour, the number of seconds in a minute
      Libraries (e.g., for calculations).
3.What are the outputs?      hours minutes seconds
  What format are they in?    integers
  How do you output them? screen
```

Test data should include at least 6 cases
0 hours
1 or more hours
0 minutes
1 or more minutes
0 seconds
1 or more seconds

Test case name (ex. "negative height", "typical conditions", etc.)	**Input data set for this test case**	**Explain why you chose this test case**

Sample solution:

```
public class Stopwatch {
	public static void main(String[] args) {
		int seconds=1456, hours, minutes, leftOver;
		final int SECONDS_PER_MINUTE=60, MINUTES_PER_HOUR=60; //use of constants necessary

		hours = seconds / (SECONDS_PER_MINUTE*MINUTES_PER_HOUR);
		leftOver = seconds % (SECONDS_PER_MINUTE*MINUTES_PER_HOUR);
		minutes = leftOver / SECONDS_PER_MINUTE;
		leftOver = leftOver % SECONDS_PER_MINUTE;

		System.out.println( seconds+" seconds equals " + hours+":"+minutes+":"+leftOver);
	}
}
```