RIT | Golisano College of Computing and Information Sciences
School of Information

Name: _____

ISTE-120
Lab 03: Calculators

Exercise 1: The Calculator Class

The first part of this lab is to write a class that simulates a calculator. Our calculator will be pretty simple. It will do addition, subtraction, multiplication, and division.

Calculators work with a single storage cell, called the accumulator, that keeps track of the current value in the calculator. This accumulator will be (the only) attribute of this class. It will be private (of course) and its type will be double. A double value in Java is one which stores a number (as opposed, for instance, a String). Further, a double can store a number with a fractional component (i.e. with digits to the right of the decimal point).

Our only constructor will be the default constructor (the one without any parameters inside the parentheses). It will initialize the accumulator to zero.

The methods will be:

- An accessor (getAccumulator)
- A method to add a number to the accumulator
- A method to subtract a number from the accumulator
- A method to multiply the accumulator by a number
- A method to divide a number into the accumulator
- A method to clear the calculator (i.e. set the accumulator to zero)
- A toString method that returns the accumulator as a String

The division method will refuse to divide 0 into the accumulator. If the user tries to do this, it will print the error message:

```
ATTEMPT TO DIVIDE BY ZERO – IGNORED
```

It will NOT change the accumulator in this case.

TWO NOTES:

1. To convert a double (the accumulator) to a String (in toString) concatenate it to the empty string, as in:

```
return "" + accumulator;
```

2. To check for a value equal to 0.0, use the form:

```
if (value == 0.0) {
   // code to execute if the value IS 0.0
}
else {
   // code to execute if the value is NOT 0.0
}
```

Consider this simple test class to understand how the calculator can be made to work.

```
public class TestCalc1 {
   public static void main(String[] args) {
      Calculator myCalc = new Calculator();
      double myWeight = 145;  // instead of 145 use your weight in pounds
      myCalc.add(myWeight);
      myCalc.multiply(0.453952);    // convert weight to KG
      System.out.println("My weight is " + myWeight + "lb = " +
         myCalc + "kg");
      }
}
```

Type this test program in and get it to work. Expand it so that you have tested all of the methods, including dividing by zero and dividing by a non-zero value. As part of your tests, write code to calculate an approximation of π using the formula:

$$\pi \approx 22/7$$

Also, calculate the number of people per square mile in New York State (NYS). Do a web search to find the population of NYS and the total area of NYS.

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 1 works correctly.

Exercise 2: Using More than One Calculator

While this calculator works, it does not easily evaluate all expressions. For example, consider the formula to calculate the slope of a line through two points (x1, y1) and (x2, y2). The slope is the difference in the y's over the difference in the x's:

$$Slope = \frac{y2-y1}{x2-x1}$$

This is tough with our calculator, because we need to calculate x2 – x1 and then divide it into y2 – y1. However, if we calculate y2 – y1 first, calculating x2 – x1 will mess up the accumulator.

One solution to this is to use two calculators, as in:

```
Calculator numerator = new Calculator();
numerator.add(y2);
numerator.subtract(y1);       // numerator now holds y2 – y1

Calculator denominator = new Calculator();
denominator.add(x2);
denominator.subtract(x1);     // numerator now holds x2 – x1

numerator.divide(denominator.getAccumulator()); // divide the denominator
                                                // into the numerator
```

Try this out with some values of (x1, y1) and (x2, y2) to see what you get. Some values and the correct answers are below. You can hard code in the values for the x's and y's.

x1	y1	x2	y2	slope
2	2	4	4	1
2	2	0	4	-1
2	3	5	2	-0.333…

Now, using what we just learned about using more than one calculator for more complex formulas, write a program, CalcBMI, that will calculate a person's Body Mass Index using the formula:

$$BMI = \frac{703*W}{H^2}$$

Where W is a person's weight (in lbs) and H is the person's height (in inches). It is OK to hard code in the values for W and H.

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 2 works correctly.

Exercise 3: Quiz Average Calculator
(this exercise may be completed outside of class and signed off at the next lab period)

This exercise is to implement a special kind of calculator. This calculator will keep track of the total of all quiz scores attained by a student, and the total number of quizzes taken, and will then return the average score for those quizzes.

The class, named 'QuizCalculator', will have attributes for:

- The student's name (a string)
- The total number of quizzes taken (initially zero)
- The sum of all the quiz scores taken so far (initially zero)

It will have one constructor, that expects the student's name as the only parameter. It will copy this into the name attribute and set the totals to zero.

It will have three methods:

- A method: `public void addScore(double newScore)`
  which will add the newScore into the sum of all quiz scores and increment the total number of quizzes taken by one.
- A virtual accessor: `public double getAverage()`
  which will return the quiz average for the student (sum of all scores / total number of quizzes). **Remember:** a virtual accessor has NO actual attribute variable, but is calculated on the fly when you get it.
- A method: `public String toString()`
  which will return the student's name and average as in:
  ```
  William Johnson quiz average: 92.50
  ```
  NOTE: the `toString` method should call the `getAverage` method to obtain the quiz average for the student.

Also write a test class with the main program in it to test this out for at least two students. Each student should have at least 3 quizzes to average. The student averages should be printed using `toString()`

Special Note
You should print the quiz averages to exactly two decimal places. To do this, you will use the `format` method of the `String` class as in:

```
String.format("%6.2f", value)
```

This returns a `String` which is the `value` formatted according to the `%6.2f` format specifier. The `value` must be a `double`. The format specifier says to format the `value` as a 6-character `String` with 2 digits to the right of the decimal point. So, if the value is a student's quiz average (as returned by `getAverage()`, it is formatted as follows:

```
D D D . D D
```

This shows a string 6 spaces long with a decimal point and two places to the right of the decimal point. The 'D's just represent digits in the string. If the value is `85.63452` the result of `String.format("%6.2f", value)` will be `" 85.63"`. Since there are only two digits to the LEFT of the decimal point, the first digit will be replaced with a space. Digits after the 1st two to the RIGHT of the decimal point will be truncated (actually, rounded to two places).

SO - the `%6.2f` is a 'picture' of what the value will look like and the value is then placed in the picture, around the decimal point.

NOTE: the 6 (6 characters) INCLUDES the decimal point.

Why 6??? Because after the decimal point (1 space) and two digits to the right of the decimal point (2 spaces) this leaves 3 spaces to the left of the decimal point. A perfect score of 100.00 requires 3 spaces to the left of the decimal point and therefore a total of 6 spaces.

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 3 works correctly.