Exercises: up to Chapter 3

This is a set of exercises based on the material in the first three chapters that don't count for credit, but (a) I'm happy to look at your solutions and (b) they are a source of extra practice - *so you should not view them as optional. (Though you might want to skip the 'tricky' and 'challenge' tasks for now.)*

For these first exercises you need to be careful about the possible effect of integer vs floating point division - remember if you divide integers, you will be doing integer division. This can have a big impact. For example 7/8 is 0! But 7.0/8.0 is 0.875 (which is fairly close to one - significant difference).

1. Write a program that reads a number representing a number of mm and convert it to inches. There are 25.4mm per inch - you need to think about the pros and cons of using integers or doubles to represent the data. Also you'll need to print out your results.

2. Calculate circumference and area of circle when user enters the radius - Google the formulae for circumference and area if you can't remember them (just typing in 'area of circle' etc. works). Remember you need to represent π - this (a) needs to be an approximation (say, 3.1415) and (b) should be represented as a *final variable. (Slight Challenge:* Java already has a 'sort-of-built-in' final variable for π so find out how to use that.)

3. Extend the circle program so it also computes the area and volume of a sphere with that radius (again Google the formula).

4. Google the formula for converting Fahrenheit to Celsius and write a program to implement it - again you'll need to read in and print out values.

5. Do the same for Celsius to Fahrenheit

6. Output the square and cube of a number, nicely formatted with the original number. For example, you can print something like this:

   The square of 2 is 4 and the cube is 8

7. Extend your square/cube program to output the square root as well - you can compute the square root of a number n in Java with `Math.sqrt(n)`.

8. Print out the sum, product, difference, absolute difference (ignoring sign), average, largest and smallest of a pair of integers entered by the user. Given two numbers a and b you can computer the maximum with `Math.max(a, b)` and the minimum with `Math.min(a,b)`.

9. Write a program that prints this:

   ```
   +–+–+
   |  |  |
   +–+–+
   |  |  |
   +–+–+
   ```

   Your program should declare two string variables, one for each kind of row, and then print them out in the right order - that is, the actual strings of characters for each type of row should only appear in your program *once*.

The next two exercises can be done more easily with some extra operations on strings. You can convert an integer `n` in Java into a string with `Integer.toString(n)`. You can extract a substring from a string in Java like this:

```
String dessert = "lemon sorbet";
String fruit = dessert.substring(0,5); //value is "lemon"
String kind = dessert.substring(6,11); //value is "sorbet"
```

The first number is the start of the substring you want to get (starting with zero) and the second is the end of the substring (again starting from zero).

The tricky bit will be dealing with the exercises below is numbers that are not fixed in length - e.g. 20 000 vs 200 000. You can find the length of a string like this (assuming `dessert` and `fruit` are the string variables from the example above):

```
int dessertLength = dessert.length(); //value is 11
int fruitLength = fruit.length();//value is 5
```

10. **Tricky:** Different countries use different symbols to format numbers. For example, the UK and US use a comma to separate thousands in a number:

    200,000

    Many other countries use a space:

    200 000

    Write a program that inputs a number and outputs it in both formats

11. **Trickier:** Extend your number so that it does decimals too - the UK and US use '.' as the decimal separator (hence decimal point) while many other countries use ',' instead. Write a program that output numbers in both formats - so 20000.3 should be 20,000.3 and 20 000,3 - you can convert a floating point number f in Java to a string with `Double.toString(f)`

    (Getting geeky, technically the UK uses '·' - a dot raised to the midline - but nobody bothers).

12. **Challenge** - compute the date of Catholic/Protestant Easter for a given year. To do this:

    Let `y` be the year (an integer)
    Let `a` be the remainder of `y` divided by `19`
    Let `b` be y divided by 100
    Let `c` be the remainder of `y` divided by `100`
    Let `d` be the `b` divided by `4`
    Let `e` be the remainder of `b` divided by `4`
    Let `g` be `8 * b + 13` divided by `25`
    Let `h` be the remainder of `19 * a + b - d - g +15` divided by `30`
    Let `j` be c divided by `4`
    Let `k` be the remainder of `c` divided by `4`
    Let `m` be `a + 11 * h` divided by `319`
    Let `r` be the remainder of `2 * e + 2 * j - k - h + m + 32` divided by `7`
    Let `n` be `h - m + 4 + 90` divided by `25`
    Let `p` be the remainder of `h - m + r + n + 19` divided by `32`
    Easter is on day `p` of month `n`

    Geeky facts - 19 appears several times in this because of the *Metonic Cycle* (look it up:-)

13. **Super Challenge** - find/work out the algorithm for computing the date of Greek Orthodox Easter and do that.

14. **Mega Challenge** - pick another worldwide religious holiday that is not on a fixed date of your choice, find/work out the algorithm for computing it, and implement it.