Exercises: up to Chapter 6

1. Write a loop that outputs the sum of all squares of numbers between 0 and 100 (remember to pick the correct kind of loop - you you know in advance how many times the loop will go around?)
2. Watch either the *Netbeans* or *Eclipse* videos (inside the Running and Installing Java folder). Redo some of the simpler examples from Exercise sheets 1 and 2 (and/or the Lab sheets if you want) so you get to learn how they work. You can do both Eclipse and Netbeans if you want, and decide which one you like best.
3. **Challenge** (sort of) - instead of Eclipse or Netbeans you might want to try IntelliJ IDEA - it's not on the lab machines but is a free download from `https://www.jetbrains.com/idea/`
4. Go back to Exercise Sheet 1 and do the Easter Challenge task (it should be straightforward for you by now). Once you've done that, extend it so that it prints out the date of Easter from 1960 to 2040. Extend it again so that the user can specify (a) the start year; (b) the end year; and (c) the interval - that is, instead of printing Easter for each year the user might be able to specify that it prints, say, Easter in 5-year steps (so instead of 2000, 2001, 2002 etc., it would print Easter for 2000, 2005, 2010). Extend your program again so that it performs 'sanity' checks on the numbers the user enters - think carefully what these should be.
5. Write a for loop that prints out the numbers 1 to 100. For all numbers that are multiples of 3 it should also print "fizz", for multiples of 5 it should print "buzz" and for multiples of both it should print "fizzbuzz". This is a common interview question for programmers and should be easy but an alarming % of "programmers" at interview can't do it.
6. Write a for loop that computes the total of the following sequence of numbers:

   1/30 + 2/29 + 3/28 + … + 30/1
7. **Tricky but not a Challenge** - you can compute the *greatest common divisor* (GCD) of two integers A and B like this:

   ```
   While the value of B is not zero:
       Set R to be the remainder of A divided by B
       Set A to be equal to B
       Set B to be equal to R
   ```

   At the end of this process, A will contain the GCD of the (original) values of A and B.
   For example:

   ```
   Let A = 12 and B = 8
   B is not zero so - R = A % B = 4; A = 8; B = 4
   B (still) not zero - R = A % B = 0; A = 4; B = 0;
   ```

   So the GCD of 12 and 8 is 4 (which is correct).
   Write a program to implement GCD.
8. **Challenge** - Credit card numbers include a check digit - the last one - which helps to check if the number is 'legal' (that is, at least could possibly be a real credit card number - it guards against data entry errors). This is a common practice - if you applied to University via UCAS, the last digit of your UCAS number was a check digit too. Check digits are calculated in different ways - for credit cards it's:

   Starting with the rightmost digit, add up all the alternate digit values. For example, if the number is:

   2234 5678 9876 5430

   starting at the rightmost 2, add up:

   2 + 4 + 6 + 8 + 8+ 6 + 4 + 0 = 38

   Take the remaining digits, double them, and add them up:

   3*2 + 5*2 + 7*2 + 9*2 + 7*2 + 5*2 + 3*2 + 2*2 = 82

   Add the two numbers together: 82 + 38 = 120.

If the last digit of the sum (110) is zero, then the number is a legal credit card number (or at least *could* be - I just made this one up).
Implement this algorithm and test it - see if it's true for any credit card you own but do NOT put your actual credit card data in the program - suggest you get your program to read in the number one digit at a time.

9. **Challenge** - An *Armstrong Number* is one in which the sum of the cubes of the digits is equal to the number itself: e.g. 371 (3^3=27,7^3=343,1^3=1; 27+343+1=371). Write a Java program to compute all the Armstrong numbers with three digits (there are four).
10. **Challenge** - The *Sieve of Eratosthenes* is an algorithm to work out which of a range of numbers is prime. Look up and understand the algorithm, and implement it in Java. (Also find out who Eratosthenes was)
11. Write a program that asks the user to enter integers and adds them to a running total until the user enters a negative number - your program should then stop and print out the total.
12. Modify your program so it works with non-integer numbers.
13. Extend your program from the exercise above so it stops when the user types anything other than a number - your program will have to be able to handle non-integer input without crashing (there are example programs, videos and things in the notes about this).
14. A roulette wheel has 37 numbers numbered from 0 to 36. Between 1 and 10, and between 19 and 28, odd numbers are red and even numbers are black. Between 11 and 18, and between 29 and 36, odd numbers are black and even numbers are red. Zero is green
    Write a program to simulate a roulette wheel - when you run your program, it should generate a random value between 0 and 36, print that out; and also print out the colour (red, black or green). HINT: to generate a random number in the range of 0 to 10:

```
import java.util.Random;
…
Random rnd = new Random();
…
int num = rnd.nextInt(9);
```

15. **Challenge** - Extend your roulette program to actually play roulette. You start with £1000 pounds and each round you bet a certain amount (an integer for simplicity). You also choose either to bet on a colour (red or black); or on an actual number. If your bet is wrong in either case you lose. If you win, and you bet on a colour you win twice your bet; if you bet on a number you win 36 times your bet. You can choose to exit the program and take your winnings (if any:-) at any time. The program will end automatically if you run out of money:-)
16. The next examples depend on the following. A way to approximate π (that is hideously inefficient) is the series:

    $\pi = 4(1/1 - 1/3 + 1/5 - 1/7 + \ldots 1/n)$

    where n is odd. Write a for loop that computes to whatever value of n you choose - it needs to be pretty big (10 000 or so) to start getting anywhere near reasonable. *Make sure you force the arithmetic to use doubles not ints or it will not work (very well).*
    Tricky but not a Challenge - The obvious way to do this is to have an if statement inside a for loop. The if statement needs to work out (a) if the value of the loop counter is odd (because you should only add/subtract a term if it is); and (b) whether to add a term or subtract a term (you might want a boolean for that which switches from true to false - true to add; false to subtract.
    More of a Challenge - work out how to avoid having to check if the value of the loop counter is odd (hint: loops don't have to count up/down in steps of 1).
    Challenge - work out how to get rid of the if statement inside the loop altogether.