# Flow Control Practical

## Objectives

The objectives of this practical session are to use `if` statements, a `switch` statement, a `while` loop and a `for` loop.

## Reference Material

This practical is based on material in the *Fundamentals* and *Flow Control* chapters.

## Overview

The first part of this practical uses a `switch` statement to determine the number of days in the current month.

The second practical uses a `while` loop to display a table of powers of 2. Then, you can repeat the exercise using a `for` loop.

## Practical

### Using the `switch` statement


1. Create a new project, exactly as you did in the previous practical. Locate the starter code under `exercises\student\Flow_Control\`. You will find a folder called `src`, under which there is a folder `qa\flow`, which correspond to package directories.
2. In IntelliJ, create the package `qa.flow`, and copy the `.java` files from the starter folder into the corresponding folder in your IntelliJ project.
3. The package contains the skeleton code of a class called `Calendar` with a single method called `main()`. We have already declared and initialised a couple of variables for you, as follows

```
byte currentMonth = 3; // a number between 0 and 11
boolean leapYear = false;
```

   A comment inside the `main()` method clearly marks where you should write your code.

4. Add a statement to display the current month.
5. Declare a variable to hold the number of days in the current month. Then, using a `switch` statement, determine the value to store in this variable. For the time being, ignore the `leapYear` flag.

   The hardest part of this exercise is remembering how many days there really are in each month, so just in case you've forgotten:

   *There are 30 days in September, April, June and November. All the rest are 31, except for February, which is 28, except in a leap year when it is 29.*

Build and test the program. Experiment with various values of `currentMonth`. What happens if you specify an invalid value such as 13?

7. Modify your code to take account of the `leapYear` flag. Rebuild and test your program.

## Practical

### Using `while` and `for` loops


1. The starter code for this practical is contained in the class `qa.flow.Numbers`.
2. The class `Numbers` has a single method called `main()`. Comments inside the `main()` method clearly mark where you should write your code.
3. Using a `while` loop, compute $2^n$ for values of n between 1 and 64 inclusive. Display the results in a table with columns for n and $2^n$ with a tab character in between. The output should look like this:

   1 2

   2 4

   3 8

   4 16

   5 32 etc etc up to 64 both columns are left aligned

4. For the time being, don't worry too much about the formatting. Do not try to find a method to do exponentiation, you don't need one just multiply repeatedly by 2.
5. Build and test the program. Does it work correctly? If not, you could run it under the debugger to find out what's going wrong.
6. Modify the loop to limit the results to 10 digits. Do this by breaking from the loop when the 'result' is bigger than the biggest 10 digit number you can hard code into the program. Remember how to define a 'long' literal. Rebuild and test the program.
7. Now change the code so that it is using a `for` loop rather than the `while` loop.
8. Modify the loop to right-align the numbers displayed in the *n* column (i.e. insert a space if the number is less than 10).
9. If you have time, modify the code in the `for` loop to format the result column as right justified. To achieve this, you'll need to insert extra spaces between the two columns in order to right-align the numbers in the second column. The number of required spaces depends on the number of digits displayed, so you could do this using a nested `for` loop. You do not need to try and create a string object (covered in later chapters) just print a single space where

necessary (several times). [Hint :- `somevariable *= 10` might come in useful.]