# Operators and Flow Control

## Overview

In this lab, you will write an application to make use of operators, conditional logic, and loops.

## Source folders

| | |
|---|---|
| Student project: | `StudentOpFlow` |
| Solution project: | `SolutionOpFlow` |

## Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Using operators
2. Using conditional logic
3. Using loops
4. Additional suggestions

## Exercise 1: Using operators

In the student project, write code to manipulate day, month, and year values using operators.

Suggestions and requirements:

- To start off with, declare 3 integers with hard-coded values for the day, month, and year.
- Use a conditional operator (`?:`) to determine if it's a leap year. A leap year is:
  (evenly divisible by 4 AND NOT evenly divisible by 100) OR
  (evenly divisible by 400)
  Use the remainder operator (%) to help you out here.
- Test whether your "is-leap-year" algorithm works for various years.

## Exercise 2: Using conditional logic

Validate the day, month, and year values. Output a single message saying whether the date is valid or not. If the date is valid, output it in the format dd / mm / yyyy.

Suggestions and requirements:

- Write an `if`-statement to validate the month (it must be 1…12).
- Write an `if`-statement to validate the year (for the sake of argument, let's say the year must be 0…2099).
- Write an `if`-statement to ensure the day is 1 or higher.
- Write a `switch`-statement to ensure the day isn't too big. The maximum value is 28, 29, 30, or 31, depending on the month (and if the month is February, whether it's a leap year).

## Exercise 3: Using loops

Refactor your application so that it gets a series of dates from the user, rather than having a single hard-coded date.

Suggestions and requirements:

- Write a `do-while` loop that allows the user to enter a series of different dates. Each time round the loop, ask the user whether he/she wants to continue; if the user says "Yes" or "yes", keep on iterating.
- Inside the loop, ask the user to enter a new day, month, and year. Use the `Scanner` class to help you here (see chapter 2 for a reminder of `Scanner`).
- For each date, validate it as you did in Exercise 2 above.

## Exercise 4 (If time permits): Additional suggestions

Improve your application so that it outputs valid dates in a format such as 25 December 2014. (Use a `switch`-statement to map the month number to the relevant month name).

When you're outputting a day, add a suffix such as *st*, *nd*, *rd*, or *th*. For example, 25th December 2014. (Use a `switch`-statement to determine when to use each suffix).

Declare a `String` variable that keeps a record of all the valid dates entered by the user. Each time the user enters a valid date, append the date (in string format) to your `String` variable. Output the `String` variable at the end of the application.