# Writing and Calling Methods

## Overview

In this lab, you will take an existing application where all the code is written in main(), and refactor it so that it makes proper use of methods.

## Source folders

Student project: StudentMethods

Solution project: SolutionMethods

## Roadmap

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

1. Refactoring the application to use methods
2. Getting user input
3. Validating user input
4. Using method overloading

## Familiarization

Open the student project, expand the student.methods package, and take a look at the code in DateProcessing.java. This code is similar to the functionality from the “Flow of Control” lab.

The main() function is quite lengthy, and performs the following tasks:

* Declares local variables to represent a day, month, and year, and initializes them with hard-coded values.
* Determines if the year is a leap year.
* Determines the number of days in the month.
* Determines the name of the month (e.g. month 1 is “January”)
* Displays all dates in a month in verbose format, such as 1st December 2017.

Make sure you’re happy with the code, then run the application and verify it works as expected.

## Exercise 1: Refactoring the application to use methods

Refactor the code in the DateProcessing class, so that some of the functionality is carved out of main() and placed into dedicated methods instead.

We suggest the following methods (remember to make the methods static so you can call them from the main() method):

* boolean calcIsLeapYear(int year)  
  Takes a year number (e.g. 2017) and returns true if a leap year or false otherwise.
* int calcDaysInMonth(int month, boolean isLeapYear)   
  Takes a month number (1…12) and a boolean indicating whether it’s a leap year, and returns the number of days in that month. For example, if the month is 2 and it’s a leap year, then there are 29 days in that month.
* String calcMonthName(int month)  
  Takes a month number (1…12) and returns the name of the month. For example, if the month is 2, the month name is “February”.
* String calcSuffix(int day)  
  Takes a day number (1…31) and returns the suffix for that day number. For example, if the day number is 1, the suffix is “st”; if the day number is 2, the suffix is “nd”, etc.

Modify main() so that it calls these methods. You should find main() is a lot easier to read after you’ve made these changes. Don’t forget to run the application to make sure it still works!

## Exercise 2: Getting user input

Enhance the application so that it asks the user to enter values for the day, month, and year (rather than simply hard-coding these values in the main() method). Use the Scanner class to simplify reading values from the keyboard.

## Exercise 3 (If time permits): Validating user input

Make the application more bullet-proof by validating the user’s input as follows:

* The year must be between 0 and 2099 inclusive.
* The month must be between 1 and 12 inclusive.
* The day must be between 1 and an appropriate upper limit inclusive. The upper limit depends on the month (i.e. 28, 29, 30, or 31).

If the user enters an invalid number, prompt the user to re-enter the value (use a do-while loop).

*Note:* There’s a certain amount of similarity in the code for validating the year, month, and day. In each case, you have to display a prompt message (e.g. “Please enter a year”), get an integer from the keyboard (e.g. using a scanner’s nextInt() method), and then wrap it all in a do-while loop to keep prompting the user if their input is invalid. To promote reuse and avoid code duplication, we suggest you write a single method as follows:

* int promptUserForNumber(String promptMessage,   
   Scanner scanner,   
   int min,   
   int max)

The method takes 4 parameters:

* promptMessage allows main() to pass in an appropriate prompt message on each call (e.g. “Please enter a year”).
* scanner allows the user to use the Scanner object first created in main(), to get input from the keyboard.
* min is the minimum value (inclusive) for validation purposes.
* max is the maximum value (inclusive) for validation purposes.

Refactor main() to call this method as appropriate to get the day, month, and year, safe in the knowledge that the input will be valid. Then run the application and verify it traps all kinds of invalid user input.

## Exercise 4 (If time permits): Using method overloading

Implement an overloaded version of the promptUserForNumber() method. This is similar to method you wrote in the previous exercise, except that it doesn’t do validation:

* int promptUserForNumber(String promptMessage,   
   Scanner scanner)

Then implement a third version of promptUserForNumber(). This is similar to before, except that it doesn’t take a Scanner parameter, and instead always gets user input from the System.in device:

* int promptUserForNumber(String promptMessage)

Consider how you can invoke these methods from main() to test your new overloaded versions of the promptUserForNumber() method.