Background and Objectives

This exercise is a synthesis to validate the core concepts you have seen so far, common to most programming languages:

- Read user input / Print output to user
- Variables and methods
- Program flow and control structures

Guess a Number rules

We will implement a simple game to have the user guess a number.

- The game starts by picking a random number between two numbers
- Then the player has to guess the right number
- If he picks the right number, the game is over and the player wins
- If he picks the wrong number, the game tells him if he is over or under the right number and the player picks another number.

Specs

`guess_number.rb`

- Implement the `#random_number` method which returns a random number between 1 and 20.
- Implement the `#right_number?` method which returns `true` if the player guesses the correct number.
- Implement the `#greater_number?` method which returns `true`if the player guess is larger than the right number, and `false`if it's smaller.

`interface.rb`

- Implement the `#too_much` method which builds a message telling the player his guess was too high.
- Implement the `#not_enough` method which builds a message telling the player his guess was too low.
- Implement the `#well_done` method which builds a message telling the player his guess was right.
- Implement the main `#play` method which runs the game from the terminal. It should work this way:

```
$ ruby lib/run_game.rb
Guess a number between 1 and 20
> 15
That's too much! Guess again.
> 8
That's not enough! Guess again.
> 12
Well done! you won!
$
```

Key learning points

- What are the different ways of looping?
- What are the different conditional structures available?

Further suggestions & resources

- You might want to use the Random class.
- When looping, you need a condition to make your game loop stop at some point.