# Test Driven Development Exercises

Learning how to develop software using TDD is often accomplished while practicing Katas. KataCatalogue has a number of different kata exercises.

Work with a partner to solve the following exercises while using the TDD approach.

## Kata String Calculator

Create a simple String Calculator with a method int add(String numbers).

### Step 1

```
The method can take 0, 1, or 2 numbers and will
return their sum. For an empty string it will return
0.
```

*Sample Output*

```
"" -> returns 0
"1" -> returns 1
"1,2" -> returns 3
```

**Hint** Begin with the simplest test case using an empty string and move to 1 then 2 numbers.

### Step 2

```
Allow the add method to handle an unknown amount of
```

```
numbers.
```

*Sample Output*

```
"1,5,7" -> returns 13
```

## Step 3

```
Allow the add method to handle new lines between
numbers (instead of commas)
```

*Sample Output*

```
"1\n2,3" -> returns 6
"3\n5\n2,4" -> returns 14
```

The input "1,\n" is not valid. A comma will not end on the line. You do not need to code for it.

## Step 4 (Bonus)

```
Support different delimters. To change a delimiter,
the beginning of the string will contain a separate
line that looks like "//[delimeter]\n[numbers...]"
```

*Sample Output*

```
//;\n1;2" -> returns 3
```

```
//!\n4!9" -> returns 13
```

# Kata Numbers to Words (Challenge)

It occurs now and then in real life that people want to write about money, especially about a certain amount of money. If it comes to cheques or contracts for example some nations have laws that state that you should write out the amount in words additionally to the amount in numbers to avoid fraud and mistakes. So if you want to transfer 745 $ to someone via cheque you have to fill out to fields:

745.00 (amount in numbers) seven hundred and forty five (amount in words)

## Step 1

Write a converter class that can convert from numbers into words.

**Test Cases to Consider**

- **1 digit numbers** (zero, three, seven)
- **2 digit numbers** (ten, fourteen, twenty-six)
- **3 digit numbers** (two hundred and nine, three hundred, four hundred and ninety-eight)
- **4 digit numbers** (three thousand and four, five thousand and twenty-six, seven thousand and one hundred and eleven)
- **5 digit numbers** (forty thousand, eighty-seven thousand and six hundred and fifty-four)
- **6 digit numbers** (five hundred thousand, eight hundred and three thousand and three hundred and eight, nine hundred and ninety-nine thousand and nine-hundred and ninety-nine)

## Step 2

Writer a converter class that goes from words back into numbers.