KataRomanNumerals

About this Kata

I wasn't there, but I believe this Kata was performed at XP2001 by Kent Beck. Here is [a video of Karl Scotland doing this Kata in Excel at agile 2008] Here is [a video of JonJagger doing this Kata in Ruby using CyberDojo]

Difficulty - Easy.

Problem Description

The Romans were a clever bunch. They conquered most of Europe and ruled it for hundreds of years. They invented concrete and straight roads and even bikinis[1]. One thing they never discovered though was the number zero. This made writing and dating extensive histories of their exploits slightly more challenging, but the system of numbers they came up with is still in use today. For example the BBC uses Roman numerals to date their programmes.

The Romans wrote numbers using letters - I, V, X, L, C, D, M. (notice these letters have lots of straight lines and are hence easy to hack into stone tablets)

The Kata says you should write a function to convert from normal numbers to Roman Numerals: eg

```
     1 --> I
     10 --> X
     7 --> VII
```

etc.

For a full description of how it works, take a look at [this useful reference website]: which includes an implementation of the Kata in javascript.

There is no need to be able to convert numbers larger than about 3000. (The Romans themselves didn't tend to go any higher)

Note that you can't write numerals like "IM" for 999. Wikipedia says: *Modern Roman numerals ... are written by expressing each digit separately starting with the left most digit and skipping any digit with a value of zero. To see this in practice, consider the ... example of 1990. In Roman numerals 1990 is rendered: 1000=M, 900=CM, 90=XC; resulting in MCMXC. 2008 is written as 2000=MM, 8=VIII; or MMVIII.*

Part II of the Kata

- Write a function to convert in the other direction, ie numeral to digit

Clues

- can you make the code really beautiful and highly readable?

  * does it help to break out lots of small named functions from the main function, or is it better to keep it all in one function?

- if you don't know an algorithm to do this already, can you derive one using strict TDD?

  * does the order you take the tests in affect the final design of your algorithm?
  * Would it be better to work out an algorithm first before starting with TDD?

- if you do know an algorithm already, can you implement it using strict TDD?

  * Can you think of another algorithm?

- what are the best data structures for storing all the numeral letters? (I, V, D, M etc)
- can you define the test cases in a csv file and use FIT, or generate test cases in xUnit?
- what is the best way to verify your tests are correct?

Suggested Test Cases

Exercise left to the reader. You could use 1999 as an acceptance test.

Comments from those who are working on this Kata

We tackled this Kata at this GothPy081007 meeting -- EmilyBache

[Here's a Scala solution], -- HansBrattberg?

Found a [F# solution] and a [tidy Ruby one] at github.

[Implementation] in ruby, java and clojure along with acceptance tests written in ruby and interacting with the command line. Would be nice with some additional languages. - FredrikRubensson

---