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 on your own to solve the following exercises while using the TDD approach.

Kata Fizz Buzz

Step 1

Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz?".

Write tests that call a single method fizzBuzz(). Given a positive integer input n, return the FizzBuzz output as a string.

Sample Output

```
1 ->  returns "1"
2 -> returns "2"
3 -> returns "Fizz"
4 -> returns "4"
5 -> returns "Buzz"
15 -> returns "FizzBuzz"
... etc up to 100
```

- Any number outside of the boundaries should return empty string -> ""

** Test Cases**

- What happens if 0 is passed in?
- What happens if 101 is passed in?
- What test cases can we come up with for the core FizzBuzz exercise?

Step 2

- A number is fizz if it is divisible by 3 or it has a 3 in it
- A number is buzz if it is divisible by 5 or it has a 5 in it

Link

Kata Potter

Once upon a time there was a series of 5 books about a very English hero called Harry. (At least when this Kata was invented, there were only 5. Since then they have multiplied) Children all over the world thought he was fantastic, and, of course, so did the publisher. So in a gesture of immense generosity to mankind, (and to increase sales) they set up the following pricing model to take advantage of Harry's magical powers.

One copy of any of the five books costs 8$. If, however, you buy two different books from the series, you get a 5% discount on those two books. If you buy 3 different books, you get a 10% discount. With 4 different books, you get a 20% discount. If you go the whole hog, and buy all 5, you get a huge 25% discount.

Note that if you buy, say, four books, of which 3 are different titles, you get a 10% discount on the 3 that form part of a set, but the fourth book still costs 8$.

Your mission is to write a piece of code to calculate the price of any conceivable shopping basket, giving as big a discount as possible.

Write tests that call a single method `double getCost(int[] books)`. Given an array of integers representing the various books purchased, calculate the cost as a decimal.

```
2 copies of the first book

2 copies of the second book

2 copies of the third book

1 copy of the fourth book

1 copy of the fifth book


(answer: $51.20)
```

Clues

- This Kata looks easy to start but there is a level that introduces complexity. However, consider that when you calculate the above basket, it isn't 5 * 8 * 0.75 + 3 * 8 *0.9. It is actually 4 * 8 * 0.8 + 4 * 8 * 0.8. The trick is to write code intelligent enough to notice two sets of four books is cheaper than a set of five and a set of 3.

Kata Roman Numerals

Step 1

The Romans were a clever bunch. They conquered most of Europe and ruled it for hundreds of years. One thing they never discovered 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 programes.

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

Sample Output

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

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.

Clues

- what are the best data structures for storing all the numeral letters? (I, V, D, M etc)

How Roman Numerals Work

Step 2

Write a function to convert in the other direction from Roman Numeral to digit.

Kata Prime Factor (OPTIONAL)

Factorize a positive integer number into its prime factors.

Use the TDD approach to write tests that call a single method factorize(). Given a positive integer input n, return its prime factors.

1 is always omitted from the result set.

Sample Output

```
2 -> returns [2]
3 -> returns [3]
4 -> returns [2, 2]
6 -> returns [2, 3]
7 -> returns [7]
8 -> returns [2, 2, 2]
9 -> returns [3, 3]
10 -> returns [2, 5]
```

Link to Wikipedia Prime Factors