Solution for Roman Numeral Generator by Murshed Dated: 08.08.2018

Problem Analysis

Looking through general numerals when numbers are multiple of tens, hundreds, thousands the letters can just added together. The exception when not multiples like a number 1995. This can represent in Roman notation as

1000 + (1000 - 100) + (100 -10) + 5
M CM XC V

The number 9, 90, 900 are less than one letter than multiple of tens and these values need pre included in data structure.

Following these rules every number at range 1 to 3999 can be represented following one and two letter combination.

M	1000	X	10
CM	900	IX	9
D	500	V	5
CD	400	IV	4
C	100	I	1
XC	90		
L	50		
XL	40		

Solution Design

I use class Tree Map<K, V > data structure to take above values as <Integer, String> and initiated with interface NavigableMap <K, V>

The interface NavigableMap<K,V> use navigation methods returning the closest matches for given search targets and this I used to solve the problem.

A NavigableMap can be accessed and traversed in either ascending or descending key order. I used descending method to access and search the map. The method used to sort the map.

numericToRomanMap.descendingMap();

Using java.util.SortedMap.entrySet() can iterate and search the map with closest matching values. getKey(), getValue() method returns Key and Values corresponding to the entry which added to the generate Roman String values.

Algorithm for Analysis and Design

---

1. Set value for Max and Min at range 1 to 3999
2. Implement Roman notation table in class TreeMap<K, V> and Navigable Map<K, V>
   2.1 Take values in Map, sort in descending order

3. If input number not in range Max Min return null
4. For each entry Iterate and search Map
   4.1 get correspond key, value
5. While number greater than or equal to Key, subtract key from number
   5.1 Add correspond value to StringBuilder
   5.2 Repeat loop until number becomes zero
   5.3 Exit
6. Return String

Using StringBuilder the value can append to generate Roman numerals as long while condition true. The toString()method I used to convert StringBuilder into equivalent Roman String literals.

Coding and Testing

I used approach fail-fast behaviour. The steps I followed

- Create interface RomanNumeralGeneratorI with generate() methods.
- Create a class RomanNumeralGenerator to implement interface

Before going further down to coding I created JUnit test class RomanNumeralGeneratorTest.java with first few test cases which obvious failed as unimplemented. Moving to RomanNumeralGenerator class started coding generate() method until test cases pass.

Test cases Assumptions

---

testMin(); testMax(); testMid(); testTooHigh(); testTooLow() and others random target assumption has been considered for reliability and fault tolerance issues.

Future Enhancement

The number limited to range 1 to 3999.

The input validation can be implemented for user choice of number to convert. This could cover negative integer, string, and other range of data types.