Assignment 03

## Java Assignment

***Same as given in assignment 02 – complete the code.***

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol Value

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.

X can be placed before L (50) and C (100) to make 40 and 90.

C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Example 1:

Input: s = "III"

Output: 3

Explanation: III = 3.

Example 2:

Input: s = "LVIII"

Output: 58

Explanation: L = 50, V= 5, III = 3.

Example 3:

Input: s = "MCMXCIV"

Output: 1994

Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

Constraints:

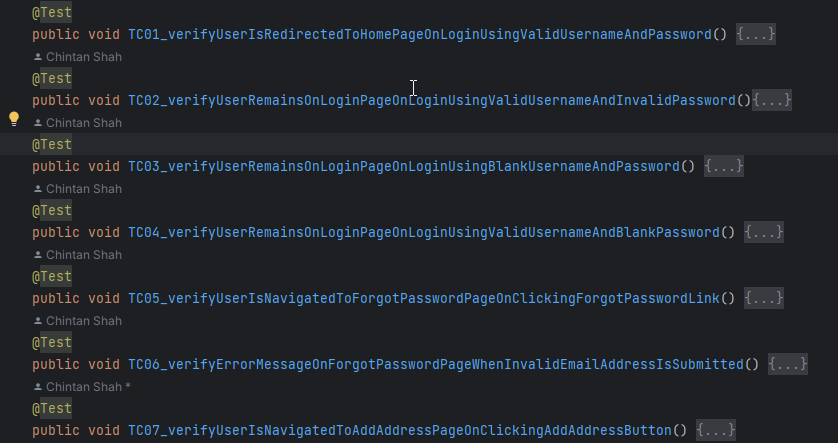
s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').

## Selenium Assignment

Write selenium code to automate following features for “TutorialsNinja” application

1. Search for “MacBook” on the top bar, go to list view and display product name, price and price after tax value.
2. Go to Desktop > Mac Menu, go to list view and display product name, price and price after tax value. In the same test case Go to “Phones & PDAs” menu, go to list view and display product name, price and price after tax value
3. Assign @Test annotation to all the test cases
4. Use “Assert” action to verify all the tests (replace all print statement with Assert commands)
5. Remove the usage and Thread.sleep and implement “Implicitly” and “Explicit” wait as required.

Note: At the end of competing assignment, you should have 12 test cases created for “Tutorials Ninja” Project with test annotations, asserts and wait strategy implemented.





## Practice code covered during lecture

1. Asserts
2. Waits

### Asserts

public void workingWithAsserts() {  
   
 //Assert - primitive data type  
 Assert.*assertEquals*(1, "1");  
 Assert.*assertEquals*(1, 1);  
 Assert.*assertEquals*('A', "A");  
   
 //Assert boolean condition  
 Assert.*assertTrue*(2 > 3);  
 Assert.*assertFalse*(2 > 3);  
   
 //Assert with failure message  
 Assert.*assertTrue*(2 > 3, "Boolean condition is false");  
   
 //Assert - arrays without order  
 String[] namesArray = {"Chintan", "Vishal"};  
 String[] names2Array = {"Vishal", "Chintan"};  
 Assert.*assertEqualsNoOrder*(namesArray, names2Array);  
  
 //Assert - lists without order  
 List<String> names = new ArrayList<>();  
 names.add("Chintan");  
 names.add("Vishal");  
 List<String> names2 = new ArrayList<>();  
 names2.add("Vishal");  
 names2.add("Chintan");  
 Assert.*assertEqualsNoOrder*(names, names2);  
}

### Waits

public void workingWithTimeouts() {  
  
 WebDriver driver = new ChromeDriver();  
 driver.manage().window().maximize();  
  
 //Default timeout settings  
 System.*out*.println(driver.manage().timeouts().getScriptTimeout());  
 System.*out*.println(driver.manage().timeouts().getPageLoadTimeout());  
 System.*out*.println(driver.manage().timeouts().getImplicitWaitTimeout());  
   
 //Set Implicit Wait  
 driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(30));  
   
 //Initialize Explicit Wait  
 WebDriverWait wait = new WebDriverWait(driver, Duration.*ofSeconds*(30));  
   
 //Explicit Wait - JQuery  
 String jQueryActiveScript = "return jQuery.active == 0";  
 wait.until(ExpectedConditions.*jsReturnsValue*(jQueryActiveScript));  
 new Select(driver.findElement(By.*id*("country"))).selectByVisibleText("India");  
  
 //Explicit Wait - Element Visible  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*id*("hiddenInput")));  
  
  
 //Initialize Fluent Wait  
 Wait<WebDriver> fluentWait = new FluentWait<WebDriver>(driver)  
 .withTimeout(Duration.*ofSeconds*(30))  
 .pollingEvery(Duration.*ofSeconds*(5))  
 .ignoring(NoSuchElementException.class);  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*id*("hiddenInput")));  
}