JUnit Testing Lab

Programmer #1: _____ Preston Ward _____

Programmer #2: _____Ishaan Thakur_____

```
/**
* Returns a string that consists of all and only the characters
* in every fifth positions (i.e., fifth, tenth, and so on) in
* the current string, in the same order and with the same case as
* in the current string. The first character in the string is
* considered to be in Position 1.
*
* @return String made of characters in every third positions in the
* current string
*/   String getEveryFifthCharacter();
```

Test ID	Test String	Test Rational	Expected Result	JUnit assertion statement
1	“Rodney, the Ram”	Test default string value	“ehm”	String every5 = defaultString.getEveryFifthCharacter(); assertTrue("ehm".equals(every5);
2	“This Is A Test String”	Test a random string	“ n”(3 spaces and an n)	String every5 = randomString.getEveryFifthCharacter(); assertTrue(" n".equals(every5);
3	“ “ (17 spaces)	Test a string of spaces	“ “(3 spaces)	String every5 = spacesString.getEveryFifthCharacter(); assertTrue(" ".equals(every5);
4	“”	Test an empty string	“”	String every5 = emptyString.getEveryFifthCharacter(); assertTrue("".equals(every5);
5	“test”	Test a string less than 5 characters	“”	String every5 = shortString.getEveryFifthCharacter(); assertTrue("".equals(every5);

```
/**
* Returns a string that consists of all and only the characters
* in the even positions (i.e., second, fourth, sixth, and so on) in
* current string, in the same order and with the same case as in
* the current string. The first character in the string is
* considered to be in Position 1.
*
* @return String made of characters in even positions in the
* current string
*/
```

`String getEvenCharacters();`

Test ID	Test String	Test Rationale	Expected Result	JUnit assertion statement(s)
1	“Test String”	Test a regular string	“etSrn”	String evenString = regularString.getEvenCharacters(); assertTrue(“etSrn”.equals(evenString));
2	“This is a string with a lot of spaces”	Test a string of lots of spaces	“his tigwt o fsae”	String evenString = lotsOfSpaces.getEvenCharacters(); assertTrue(“his tigwt o fsae”.equals(evenString));
3	“ “ (13 spaces)	Test a string of only spaces	“ “(6 spaces)	String evenString = onlySpaces.getEvenCharacters(); assertTrue(“ ”.equals(evenString));
4	“”	Test an empty string	“”	String evenString = emptyString.getEvenCharacters(); assertTrue(“”.equals(evenString));
5	“a”	Test a string less than 2 characters in length	“”	String evenString = lessThanTwo.getEvenCharacters(); assertTrue(“”.equals(evenString));

```
/**Returns the # of characters that are digits in the string
*  if two (and no more than two) digits appear side by side.
* @return Number of double-digits in the current string
*/
```

`int countDoubleDigits();`

Test ID	Test String	Test Rationale	Expected Result	JUnit assertion statement(s)
1	“11 and 12”	Test 2 two-digit numbers	1	int numDoubles = twoNumbers.countDoubleDigits(); assertTrue(1 == numDoubles);
2	“11123 345677 8990”	Test a string of numbers	3	int numDoubles = justNumbers.countDoubleDigits(); assertTrue(3 == numDoubles);
3	“00000 00”(7 zeros)	Test numbers in a row	0	int numDoubles = sameInARow.countDoubleDigits(); assertTrue(0 == numDoubles);
4	“00 00 00 00”	Test pairs of numbers separated by a comma	4	int numDoubles = commaSeparatedPairs.countDoubleDigits(); assertTrue(4 == numDoubles);
5				

```
/** Replace all occurrence of a single zero (0) with the string
  * "Go VCU" in the current string,
  * and all occurrence of a double zero (00) with the string "CS@VCU"
  */
 void ramifyString();
```

Test ID	Test String	Test Rationale	Expected Change to String variable	JUnit assertion statement(s)
1	“0 and 00 and 000”	Test as expected	“Go VCU and CS@VCU and 000”	String expectedString = “0 and 00 and 000”; expectedString.ramifyString(); assertTrue(“Go VCU and CS@VCU and 000”.equals(expectedString));
2	“00000”	Test only zeros	“00000”(no change)	String onlyZeros = “00000”; onlyZeros.ramifyString(); assertTrue(“00000”.equals(onlyZeros));
3	“0 00 0”	Test space-separated zeros	“Go VCU CS@VCU Go VCU”	String spaceSeparated = “0 00 0”; spaceSeparated.ramifyString(); assertTrue(“Go VCU CS@VCU Go VCU”.equals(spaceSeparated));
4				
5				

```
/** Replace the _individual_ digits in the current string, between
 * startPosition and endPosition (included), with the corresponding
 * Roman numeral symbol(s). The first character in the string is
 * considered to be in Position 1. Digits are converted individually,
 * even if contiguous, and digit "0" is not converted (e.g., 460 is
 * converted to IVVI0). In case you are not familiar with Roman
 * numerals, see https://en.wikipedia.org/wiki/Roman_numerals
 *
 * @param startPosition   first character to consider
 * @param endPosition     last character to consider
 * @throws MyIndexOutOfBoundsException
 *            If either "startPosition" or "endPosition" are out of
 *            bounds (i.e., either less than 1 or greater then the
 *            length of the string)
 * @throws IllegalArgumentException
 *            If "startPosition" > "endPosition" (both in bounds)
 */
void convertDigitsToRomanNumeralsInSubstring(int startPosition, int
endPosition) throws MyIndexOutOfBoundsException, IllegalArgumentException;
```

Test ID	Test String	Test Rationale	Expected Change to String variable	JUnit assertion statement(s)
1	“I am 19 years old”(1,17)	Test the entirety of a normal string	“I am IIX years old”	String normalString = “I am 19 years old”; normalString.convertDigitsToRomanNumeralsInSubstring(1,17); assertTrue(“I am IIX years old”.equals(normalString));
2	“I am 19 years old”(7,17)	Test a part of a normal string	“I am 1IX years old”	String restrictedString = “I am 19 years old”; restrictedString.convertDigitsToRomanNumeralsInSubstring(7,17); assertTrue(“I am 1IX years old”.equals(normalString));

3	“1234567890”(1,10)	Test a string of just numbers	“IIIIIIIVVVIVIIVIIIIX0”	String numberString = “1234567890”; numberString.convertDigitsToRomanNumeralsInSubstring(1,10); assertTrue(“IIIIIIIVVVIVIIVIIIIX0”.equals(numberString));
4				
5				