**Задание 1.**

Необходимо написать и протестировать 7 функций для валидации, конвертации и генерации телефонных номеров.

**validateRawNumber(String rawNumber): boolean** – метод валидации телефонного номера в «сыром виде». Телефонный номер в сыром виде есть строка из десяти десятичных цифр, например, “8005553535“. Возвращает true, если номер валидный, иначе false.

public class Main {  
 // validateRawNumber - валидация телефонного номера в сыром виде  
 // вход: телефонный номер в сыром виде  
 // выход: true, если валидный, иначе false  
 // валидный телефонный номер в сыром виде = это строка из десяти цифр, например "0055530203"  
 public static boolean validateRawNumber(String rawNumber) {  
 //int num = Integer.parseInt(rawNumber);  
// String regex = "\\d{10}";  
// return rawNumber !=null && rawNumber.length()==10 && rawNumber.matches(regex);  
 return rawNumber != null && rawNumber.matches("^[0-9]{10}$"); // by Civova C.  
 }

**validateFormattedNumber(String formattedNumber): boolean** – метод валидации телефонного номера в «форматированном виде». Телефонный номер в форматированном виде есть строка из десяти десятичных цифр, которые разделены в определенных местах дефисом, а также имеют код РФ в начале “+7” (7 – это одиннадцатая цифра в строке), например, “+7‑800‑555-35-35“. Возвращает true, если номер валидный, иначе false.

// validateFormattedNumber - валидация телефонного номера в форматированном виде  
 // вход: телефонный номер в форматированном виде  
 // выход: true, если валидный, иначе false  
 // валидный телефонный номер в форматированном виде, это строка из 10 цифр + код РФ и дефисы в нужных местах  
 // например "+7-005-553-02-03"  
 public static boolean validateFormattedNumber(String formattedNumber) {  
 return formattedNumber != null && formattedNumber.matches("^\\+7-[0-9]{3}-[0-9]{3}-[0-9]{2}-[0-9]{2}$");  
// String regex = "^\\+7-[0-9]{3}-[0-9]{3}-[0-9]{2}-[0-9]{2}$";  
// return formattedNumber !=null && formattedNumber.length()==16 && formattedNumber.matches(regex);  
 }

**convertRawToFormatted(String rawNumber): String** – преобразование «сырого» телефонного номера в «форматированный». Например, номер “8005553535“ будет преобразован в “+7‑800‑555-35-35“. Возвращает полученный номер в форматированном виде. В случае, если входной номер не валидный, возвращать null.

// convertRawToFormatted - преобразование телефонного номера в сыром виде в форматированный вид  
// вход: телефонный номер в сыром виде  
// выход: этот же телефонный номер в форматированном виде, если входной номер не валидный, то вернуть null  
// например: "0055530203" -> "+7-005-553-02-03"  
public static String convertRawToFormatted(String rawNumber) {  
 if(!*validateRawNumber*(rawNumber))  
 {  
 return null;  
 }  
 StringBuilder stringBuilder = new StringBuilder(rawNumber);  
 stringBuilder.insert(0,"+")  
 .insert(1,"7")  
 .insert(2,"-")  
 .insert(6,"-")  
 .insert(10,"-")  
 .insert(13,"-");  
 return stringBuilder.toString();  
}

**convertFormattedToRaw(String formattedNumber): String -** преобразование «форматированного» телефонного номера в «сырой». Например, номер “+7‑800‑555-35-35“ будет преобразован в “8005553535“. Возвращает полученный номер в форматированном виде. В случае, если входной номер не валидный, возвращать null.

// convertFormattedToRaw - преобразование телефонного номера в форматированном виде в сырой вид  
// вход: телефонный номер в форматированном виде  
// выход: этот же телефонный номер в сыром виде, если входной номер не валидный, то вернуть null  
// например: "+7-005-553-02-03" -> "0055530203"  
public static String convertFormattedToRaw(String formattedNumber) {  
 if (!*validateFormattedNumber*(formattedNumber)){  
 return null;  
 }  
 return formattedNumber  
 .substring(3)  
 .replaceAll("-","");  
}

**generateRawNumber(Random random): String** – генерация рандомного номера в сыром виде. Результат должен быть валидным номером в сыром виде.

// generateRawNumber - генерация случайного телефонного номера в сыром виде (должен быть валидный)  
public static String generateRawNumber(Random random) {  
 StringBuilder sb = new StringBuilder();  
 for (int i=0; i<10; i++){  
 sb.append(random.nextInt(0,9));  
 }  
 return sb.toString();  
}

**generateFormattedNumber(Random random): String -** генерация рандомного номера в форматированном виде. Результат должен быть валидным номером в форматированном виде.

// generateFormattedNumber - генерация случайного телефонного номера в форматированном виде (должен быть валидный)  
public static String generateFormattedNumber(Random random) {  
 String rawNumber = *generateRawNumber*(random);  
 StringBuilder sb = new StringBuilder(rawNumber);  
 sb.insert(0,"+")  
 .insert(1,"7")  
 .insert(2,"-")  
 .insert(6,"-")  
 .insert(10,"-")  
 .insert(13,"-");  
 String str = sb.toString();  
 return sb.toString();

}

**generateNumbers(int count, Random random): String[][]** – генерация множества радномных номеров в количестве count-штук. Сгенерированные номера записываются в двумерный массив, причем в первой строке массива располагаются номера в сыром виде, а во второй строке на соответствующих позициях располагаются те же самые номера в форматированном виде – т.е. в каждом столбце 2 одинаковых номера в разном виде. Все номера должны быть валидными.

// доп функция принимает строку - возвращает форматируемую строку  
// generateFormattedNumber - генерация случайного телефонного номера в форматированном виде (должен быть валидный)  
public static String generateFormattedNumberString(String rawNumber) {  
 //String rawNumber = generateRawNumber(random);  
 StringBuilder sb = new StringBuilder(rawNumber);  
 sb.insert(0,"+")  
 .insert(1,"7")  
 .insert(2,"-")  
 .insert(6,"-")  
 .insert(10,"-")  
 .insert(13,"-");  
 String str = sb.toString();  
 return sb.toString();  
}

// generateNumbers - генерация случайных телефонных номеров  
// вход: count - кол-во номеров для генерации  
// выход: двумерный массив сгенерированных номеров, в первой строке которого номера в сыром виде,  
// а во второй строке соответствующие им номер в форматированном виде  
public static String[][] generateNumbers(int count, Random random) {  
  
 // Создаём массив двумерный String  
 // [] - вводит пользователь(кол-во строк)  
 // [2] - фиксированное значение (кол-во столбцов)  
 // String [][] mass = new String[][] {{"0","1","2"},{"1","2"}};  
 //String[][] mass = null;  
 String[][] mass = new String[2][count];  
 for (int i = 0; i < count; i++) {  
 String rawNumberOne = *generateRawNumber*(random);  
 String rawNumberTwo = *generateFormattedNumberString*(rawNumberOne);  
 mass [0][i] = rawNumberOne;  
 mass [1][i] = rawNumberTwo;  
 }  
 return mass;  
}

Написать сценарии тестирования всех процедур в коде.

Введите ко-во массивов: 20

[[7120377844, 8612701301, 8153473701,

5258265774, 7275876788, 6512752171,

4216368733, 7810264442, 4402470856,

7365704003, 0278767735, 0521068608,

4437473348, 4577265355, 5023063071,

6068400861, 6787726043, 5128014427,

2584607116, 7521540245],

[+7-712-037-78-44, +7-861-270-13-01, +7-815-347-37-01,

+7-525-826-57-74, +7-727-587-67-88, +7-651-275-21-71,

+7-421-636-87-33, +7-781-026-44-42, +7-440-247-08-56,

+7-736-570-40-03, +7-027-876-77-35, +7-052-106-86-08,

+7-443-747-33-48, +7-457-726-53-55, +7-502-306-30-71,

+7-606-840-08-61, +7-678-772-60-43, +7-512-801-44-27,

+7-258-460-71-16, +7-752-154-02-45]]

**Задание 2.**

Необходимо написать и протестировать 4 функции для работы с IPv4-адресами

**validateIPv4(String ipv4String): boolean –** валидация ipv4-адреса, возвращает true если переданная строка является валидным ipv4-адресом, иначе false. Валидным ipv4-адресом считается строка, которая состоит из четырех чисел в диапазоне от 0 до 255, разделенных между собой точками, например – “178.27.0.1“.

**generateRandomIPv4(Random random): String –** генерация случайного ipv4-адреса. Сгенерированный адрес должен быть валидным.

**extractAllIPv4(String text): String[] –** процедура принимает на вход строк с текстом, находит в данном тексте все валидные ipv4-адреса, извлекает их и возвращает в виде массива строк без повторений.

**generateRandomText(int ipv4Count, double ipv4Percentage, Random random): String –** процедура генерации текста с ipv4-адресами. Сам текст содержит ipv4Count случайных ip-адресов, а также случайные последовательности букв, цифр, пробелов и знаков препинания. Длина текста зависит от кол-ва ipv4-адресов в нем – на вход передается процент ipv4-адресов от общей длины текста (примерно) – дробное число от 0 до 1 (не включая границы).

Т.е. вызов **generateRandomText(10, 0.5)** будет означать, что требуется сгенерировать текст, 50% символов которого будут составлять ipv4-адреса в количестве десяти штук.

Протестировать методы.