This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.

(a) A **number group** represents a group of integers defined in some way. It could be empty, or it could contain one or more integers.

Write an interface named `NumberGroup` that represents a group of integers. The interface should have a single `contains` method that determines if a given integer is in the group. For example, if `group1` is of type `NumberGroup`, and it contains only the two numbers -5 and 3, then `group1.contains(-5)` would return `true`, and `group1.contains(2)` would return `false`.

Write the complete `NumberGroup` interface. It must have exactly one method.

**Testing code link:**
https://www.jdoodle.com/a/26nx
(b) A range represents a number group that contains all (and only) the integers between a minimum value and a maximum value, inclusive.

Write the Range class, which is a NumberGroup. The Range class represents the group of int values that range from a given minimum value up through a given maximum value, inclusive. For example, the declaration

```java
NumberGroup range1 = new Range(-3, 2);
```

represents the group of integer values -3, -2, -1, 0, 1, 2.

Write the complete Range class. Include all necessary instance variables and methods as well as a constructor that takes two int parameters. The first parameter represents the minimum value, and the second parameter represents the maximum value of the range. You may assume that the minimum is less than or equal to the maximum.
(c) The `MultipleGroups` class (not shown) represents a collection of `NumberGroup` objects and is a `NumberGroup`. The `MultipleGroups` class stores the number groups in the instance variable `groupList` (shown below), which is initialized in the constructor.

```java
private List<NumberGroup> groupList;
```

Write the `MultipleGroups` method `contains`. The method takes an integer and returns `true` if and only if the integer is contained in one or more of the number groups in `groupList`.

For example, suppose `multiple1` has been declared as an instance of `MultipleGroups` and consists of the three ranges created by the calls `new Range(5, 8)`, `new Range(10, 12)`, and `new Range(1, 6)`. The following table shows the results of several calls to `contains`.

<table>
<thead>
<tr>
<th>Call</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>multiple1.contains(2)</code></td>
<td>true</td>
</tr>
<tr>
<td><code>multiple1.contains(9)</code></td>
<td>false</td>
</tr>
<tr>
<td><code>multiple1.contains(6)</code></td>
<td>true</td>
</tr>
</tbody>
</table>
Complete method `contains` below.

```java
/**
 * Returns `true` if at least one of the number groups in this multiple group contains `num`;
 * `false` otherwise.
 */
public boolean contains(int num)
```