ISC 233 Warmup: Finite Language Class

Develop a class `Language` for processing finite formal languages, supporting union, concatenation, and *n*-closure.

Definitions. A *language* is a set of strings. A *set* is a collection of elements where no two elements are equal; we provide a class `SET.java` that enforces this property. The *union* of two languages is the set of all strings that are in either or both of the two languages. The *concatenation* of two languages is the set of all strings that can be formed by appending a string from the second language to a string from the first language. The *n-closure* of a language is the set of all strings that can be formed by concatenating *n* strings from the language. Here are some examples, using RE notation:

union:	`a|abc = { a, abc }`
concatenation:	`(b|bc|bcd)(cd|d) = { bccd, bcd, bcdcd, bcdd, dd }`
2-closure:	`(e|ef){2} = { ee, eef, efe, efef }`

Your task. Make sure that you have downloaded the files `Language.java` and `SET.java` as per the online instructions. `Language.java` is a client of the `SET` data type, which takes care of maintaining sets of *distinct* elements (adding a string to a set of strings that is already contains that string has no effect, as desired). Add code to `Language.java` as indicated within the file to implement the `concatenate()` method and the `closure()` method. We have provided implementations of the constructors and the `union()` and `toString()` method to help you get started.

Example. Note that `Language.java` has a `main()` client to test your methods by printing out the strings in the languages `a|abc`, `(b|bc|bcd)(cd|d)` and `(e|ef){2}`. Your program must behave as follows (the strings on each line can be in any order):

```
% java-introcs Language
 a abc
 bccd bcd bcdcd bcdd bd
 ee eef efe efef
```

Note that the string `bcd` appears only once in the second language, even though it can be formed either by concatenating `b` and `cd` or by concatenating `bc` and `d`.

Restrictions. Also as indicated within the file, you must use only a single instance variable `language` which is `final`. In other words, your `Language` class has to be *immutable*—the invoking `Language` object cannot change during a union, concatenate, or closure operation.

Hint 1. To implement these methods, you will need to use Java's for-each loop. See the provided `toString()` and `union()` methods for examples of using a for-each loop with `SET<String>`.

Hint 2. There are many methods in `SET.java` but the only ones you will need to use for this exam are: the constructor, the `add()` method, and for-each loops (see Hint 1).

Food for thought. Why not just use an array as the underlying data structure?