Leet, also known as *1337*, is an Internet sociolect and a modification of written (in most cases English) text. Basically, some cool hacker guy from 1980 would never write “cool program” but instead “k3wl pr0gr@m”. For background information, take a look at the Wikipedia article about Leet.

Your task is to write a `Leet` class that contains methods to translate normal language into *Leetspeek*, as described in the following steps. Save your class to the file `Leet.java`.

Step 1: Implement `Leet.toLeet(…)`

Implement the method

```
public static String toLeet(String normal)
```

which translates the given string `normal` into Leet by performing the following transformations. Note that the order in which the conversions are performed is important.

Normal	Leet
elite	1337
cool	k3wl
!	!!!11
ck	xx
ers	0rz
er	0rz
en	n
e	3
t	7
o	0

Normal	Leet
a	@

Although there are plenty of other possible conversions, our lesson requires to implement only the ones in the table above.

Hint: How to replace strings

Java's `String` class provides the `replace(…)` method with the following signature:

```
public String replace(CharSequence target, CharSequence replacement)
```

When called on a `String` object, it returns a new `String` object in which all occurrences of `target` are replaced with `replacement`. (Don't be confused by `CharSequence` – this allows for the method to not only operate on `String` objects, but also on other string-like objects.) For example:

```
String original = "I am such a cool hacker!";
String modified = original.replace("cool", "crappy");
```

After that, the `modified` string contains the value `I am such a crappy hacker!`. The value of the `original` string is unchanged and still contains `I am such a cool hacker!`.

Step 2: Implement `Leet.allToLeet(…)`

Implement the method

```
public static String[] allToLeet(String[] normals)
```

which converts all elements of the given `String` array `normals` to Leet and returns them in a *new* `String` array. Use a `for` loop to convert the elements of `normals`. Make use of the `toLeet(…)` method that you developed in the first step for every single conversion.

Hint: Arrays

An array is a container for several values of the same type. It has a fixed size that must be specified at creation time. For example, the following code creates a new array that can hold two `String` objects:

```
String[] myArray = new String[2];
```

Created this way, the elements of the array will initially hold only `null` values.

Each array element can be accessed through its index. The first element has the index `0` (zero). You can write to the array:

```
myArray[0] = "First element";
myArray[1] = "Second element";
```

And you can read from the array:

```
System.out.println(myArray[0]);
```

This outputs

```
First element
```

on the console.

Reading or writing beyond the size of the array results in an `ArrayIndexOutOfBoundsException`: For example, reading from or writing to `myArray[-1]` or `myArray[3]` will fail, because only `0` and `1` are valid indexes for an array of length 2.

The length of an array can be accessed via its `length` attribute, e.g., `myArray.length`.

Hint: `for` loops

The syntax of the `for` loop in Java is the same as in any other C-like programming language:

```
for (<initialization>; <termination>; <increment>) {
    <statement(s)>
}
```

For example:

```
for (int i = 0; i < 5; i++) {
    System.out.println("for loops are awesome");
}
```

The statement in the loop body will be executed as long as `i` is smaller than 5. Starting at 0, the variable `i` is incremented by 1 after the statement inside the loop has been executed (`i++` is a shortcut for `i = i + 1`). Therefore, the `println(…)` method is executed 5 times and the output is:

```
for loops are awesome
for loops are awesome
for loops are awesome
for loops are awesome
for loops are awesome
```

Step 3: Write a `main(…)` method (optional)

Write another class which will hold the `main(…)` method. Within this method, create a `String` array with two elements and put the following `String` literals into it:

- `"We are elite hackers!"`

- `"Informatiker machen coole Sachen!"`

Next, translate all elements of that array into Leet, using the `allToLeet(…)` method you developed in the second step. Print the converted strings to the console. You should get the following output:

```
W3 @r3 l337 h@xx0rz!!!11
Inf0rm@7ik0rz m@chn k3wl3 S@chn!!!11
```

Hint: String concatenation

In Java, the plus operator (`+`) is *overloaded* and also works on strings. You can use it to *concatenate* strings, i.e., put multiple strings together.

For example, the code

```
String name = "Bob";
System.out.println("My name is " + name + ".");
```

results in

```
My name is Bob.
```

This even works for concatenation of strings and non-string values such as integers:

```
int age = 42;
System.out.println("I am " + age + " years old.");
```

Java automatically converts the integer value to a `String` object before it concatenates it with the other two strings.