When interacting with an automated dialogue system, the caller is often required to provide his phone number. Phone numbers may be uttered in many ways with different digit groupings (e.g. 2106930664 may be uttered as "210 69 30 6 6 4" or "210 69 664" etc). Furthermore, speech input may cause ambiguities. For example, if the caller says "twentyfive" this could be transcribed as "25" or "20 5".

The application you are asked to develop deals with some of these issues.

Java Assignment: Natural Numbers Interpretation

Implement a Java application that will handle natural numbers interpretation.

The application should ask the user for a number, and then present the identified number back to the user.

The application should accept as input a sequence of numbers separated by a space char. Each number in the input sequence may be up to a three digit number. This input represents recognized numbers that a user may pronounce. For example, possible inputs may be:

input example: 30 2 5 58

input example: 2 10 69 30 6 60 4

input example: 6 97 400 23 7 40 5

Then the application should identify and print the input number.

For example, if input is: 30 2 5 58, output should be: 302558

If input is: 2 10 69 30 6 60 4, output should be: 21069306604

It should be a java standalone application, running on a GUI or just from the command prompt.

***BASE LEVEL*: Phone number validation**

The application should state if the output number is a valid Greek telephone number.

Assume that valid Greek phone numbers may have 10 or 14 digits. If they have 10 digits, they must start with '2' or '69'. If they have 14 digits, the must start with '00302' or '003069'.

For example, if input is: 30 2 5 58, output should be: 302558 [phone number: INVALID]

If input is: 2 10 69 30 6 6 4, output should be: 2106930664 [phone number: VALID]

If input is: 2 10 69 30 6 60 4, output should be: 21069306604 [phone number: INVALID]

If input is: 0 0 30 69 74 0 9 22 52, output should be: 00306974092252 [phone number: VALID]

***ADVANCED LEVEL*: Identify natural number ambiguities**

The application should check for possible ambiguities in number spelling.

For example if the input sequence contains '... 20 5...' the result number may either be '...205...' or '...25...'. Also if the sequence contains '...75...', the result number may be: '...705...' or '...75...'

The application should identify these ambiguities, and return any possible number interpretations.

For example:

If input is: 2 10 6 9 30 6 6 4, then output should be:

```
Interpretation 1: 2106930664    [phone number: VALID]
Interpretation 2: 210693664     [phone number: INVALID]
```

If input is: 2 10 69 30 6 6 4, then output should be:

```
Interpretation 1: 2106930664    [phone number: VALID]
Interpretation 2: 210693664     [phone number: INVALID]
Interpretation 3: 2106093664    [phone number: VALID]
Interpretation 4: 21060930664   [phone number: INVALID]
```

If input is: 0 0 30 69 700 24 1 3 50 2, then output should be:

```
Interpretation 1: 0030697002413502     [phone number: INVALID]
Interpretation 2: 003069700241352      [phone number: INVALID]
...
Interpretation n: 00306972413502       [phone number: VALID]
...
Interpretation m: 00306097241352       [phone number: INVALID]
```