Hola ¿Cómo estas? Le queriamos informar que por temas de tiempo y cuestiones externas, no pudimos completar la segunda parte. Lo que si hicimos fue pensarlo. Entonces aquí le dejo nuestro planteo con respecto a la implementación de la segunda part.

Segunda parte: SentenceFinderByPrefix

El stack, en el cual deberá buscar el prefix, será seteado por el usuario, por ende la clase SentenceFinderByPrefix tendra un mensaje de clase el cual podria ser:

`#withStack:`

Y el mensaje inicialize el SentenceFinderByPrefix con el stack dado por el usuario.

Mensaje find:

El mensaje find recive un argumento con el nombre de `aPrefix`. Se preguntaria si `aPrefix` es nulo o si contiene espacios vacios. Esto se haría con ifNil y con un mensaje de la clase String `#findString: aString`. En el caso que `#findString: aString` devuelva distinto de 0, significa que dicho aPrefix se encuentra dentro del string.

En el caso que `aPrefix` sea nulo. Se enviara un error llamado `prefixCanNotBeNull`. Y si contiene espacios vacios, el mensaje de error se llamaria `prefixCanNotContainEmptyCharacters`. Estos dos mensajes se encontrarian en la categoria de errors, de la parte de **instancia**. Esto es porque al instanciar el objeto de `SentenceFinderByPrefix` no se le setea un prefijo, el prefijo varía segun lo que necesite buscar el usuario.

En el caso que el `aPrefix` sea correcto, se invocaria un mensaje llamado `findStringWithPrefix: aPrefix addIn: matchedSentences` (matchedSentences es un OrdererCollection en el cual se agregaran las oraciones encontradas). El cual sería una funcion recursiva que haría lo siguiente:

```
findStringWithPrefix: aPrefix addIn: matchedSentences
 |string|
 (self stack isEmpty) ifTrue: [
    ^false.
```

```
    ].

    string <- self stack pop.
    (string findString: aPrefix) ~= 0 ifTrue : [
       self matchedSentences add: string.
    ]
    self findStringWithPrefix: aPrefix addIn: matchedSentences
    self stack push: string
```

Entonces, con ese algoritmo, el orden del stack se mantiene intacto y en matchedSentences se fueron agregando las oraciones que coincidan con `aPrefix`.

Entonces, el motodo de find: seria el siguiente:

```
  find: aPrefix
    |matchedSentences|
    (aPrefix isNil) ifTrue: [
       ^self error: self prefixCanNotBeNull.
    ]

    (aPrefix findString: '') ifTrue: [
       ^self error: self prefixCanNotContainEmptyCharacters.
    ]
    matchedSentences <- OrderedCollection new.
    self findStringWithPrefix: aPrefix addIn: matchedSentences
    ^matchedSentences.
```

Tests

Los test contemplarian los siguientes casos:

1. Buscar un prefijo nulo, devuelve el error de prefixCanNotBeNull
2. Buscar un prefijo con espacios vacios, devuelve el error prefixCanNotContainEmptyCharacters

3. Buscar un prefijo válido, en un stack nulo o vacio, devuelve un OrderedCollection vacio
4. Buscar un prefijo válido, en un stack con elementos, devuelve un OrderedCollection vacio o no dependiendo del prefix. Y que devuelva la oración correcta