Ejercicio 1.

Implementa el siguiente método:

```
public static String traducir(String textoCastellano,
        Map<String,String> map);
```

, teniendo en cuenta que:

- La clave en el diccionario es la palabra en castellano y el valor es su traducción al inglés.
- El método traducir devuelve una cadena con la traducción al inglés, palabra a palabra, de la cadena *textoCastellano*.
- Si una palabra no se encuentra en el diccionario, deberá sustituirla por “<error>” en la cadena resultante.

Ejercicio 2.

Usando la interfaz *Map*, y disponiendo de la clase *TablaHash* que la implementa, diseña un programa que lea un texto por teclado y devuelva el número de palabras distintas que hay en dicho texto junto con la frecuencia de aparición de cada una de ellas.

Nota: la frecuencia se calcula como el número de veces que aparece la palabra en el texto dividido entre el número total de palabras del texto

Ejercicio 3.

La Policía quiere utilizar una *TablaHash* para poder consultar eficientemente el dueño de un coche determinado.

- Los coches se identifican por su matrícula, compuesta por un número (entero) y una secuencia de letras (String). Es importante mantener estos dos atributos por separado para acelerar otras posibles operaciones.
- Del dueño del coche sólo nos interesa saber su nombre.

Indica de qué tipo sería la *TablaHash* y diseña las clases que sean necesarias.

Ejercicio 4.

Inserta los enteros 9, 7, 3, 17, 18, 16, 12, 10, 22 en una tabla hash enlazada de tamaño 5 y con la función de dispersión H(x) = x.

a) Sin hacer *rehashing,* dibuja la tabla resultante y calcula su FC.

b) ¿Y si hiciéramos *rehashing* cuando el FC > 1.5, aumentando la capacidad de la tabla a 11?

Ejercicio 5.

Se quiere saber las entradas de un *Map*, implementado mediante una *TablaHash*, cuyo valor es igual a uno dado.

Se pide: implementar en la clase *TablaHash* un nuevo método que devuelva una *ListaConPI* con las claves de las entradas cuyo valor sea igual a uno dado.

Ejercicio 6.

Diseña un método en la clase *TablaHash* que reciba otro *Map* como parámetro y devuelva una nueva que sea la intersección de ambas (es decir, que contenga los elementos que están en ambos *Maps*):

```
public Map<C,V> interseccion(Map<C,V> otro);
```