Explicad mínimamente el código de vuestra solución, especialmente el de los métodos más complejos.

Como siempre, definid funciones auxiliares para que el código sea entendible y manejable y escoged nombres adecuados para las variables.

Problema 1 (7 puntos)

Apartado a: La clase `WordCounter` (2 puntos)

Diseña e implementa una clase denominada `WordCounter` para representar el contador asociado a una palabra

- `new WordCounter(String word)`
  - **tenemos garantizado que word no es null ni la palabra vacía**
  - constructor que crea una instancia de `WordCount` para la palabra `word` y con valor del contador igual a `1`.
- `String getWord()`
  - devuelve la palabra asociada al objeto receptor
- `int getCounter()`
  - devuelve el valor del contador asociado al objeto receptor
- `void increment()`
  - incrementa el valor del contador en una unidad

Apartado b: La clase `WordStats` (5 puntos)

La clase `WordStats` servirá para definir una colección extensible de `WordCounter`s.

- `new WordStats(int initialSize)`
  - **tenemos garantizado que `initialSize` es positivo**
  - se crea un array de `WordCounter` con tamaño para `initialSize` contadores
- `boolean countWord(String word)`
  - **tenemos garantizado que word no es null ni la palabra vacía**
  - cuenta una nueva aparación de la palabra `word`
    - si dicha palabra ya está en la colección, incrementa su contador
    - si no, la añade con contador `1`
  - si no hay espacio suficiente, se doblará el tamaño actual
  - devuelve `true` si la palabra no existía y `false` en caso contrario
- `int getCounter(String word)`
  - devuelve el contador asociado a la palabra `word` o `0` en caso de que ésta no exista en la colección

- `String max()`
    - devuelve la palabra con un contador más alto dentro de la colección
    - en caso de haber varias, cualquiera de ellas es válida
    - si la colección está vacía, devuelve `null`

Problema 2 (3 puntos)

El programa principal pedirá un texto y deberá escribir como resultado **la palabra que aparece más veces** (o una de ellas en caso de empate) **o bien que el texto no contiene palabra alguna**.

Como separador entre palabras solamente debéis considerar espacios, comas y puntos.

Una palabra como un nombre de usuario (gimeno67), que contiene algún dígito, no es una palabra válida como tampoco lo son los números (no hace falta que consideréis el signo).

```
Frase: Este enunciado de gimeno67 era un mal enunciado con errores.
La palabra que aparece más veces es “enunciado”.
```

Recordad que, para simplificar el programa principal, haremos que la instancia de `WordStats` sea variable de instancia del programa principal:

```
public class WordStatsProgram extended CommandLineProgram {

    private static final int INITIAL_SIZE = 28;
    private WordStats stats;

    public void run() {
        this.stats = new WordStats(INITIAL_SIZE);
        ???
    }

    ???
}
```

- class **CommandLineProgram**
  - String **readLine**(String message)
  - void **println**(String str)/ void **println**(int n) / void **println**(double d)
  - void **print**(String str) / void **print**(int n) / void **print**(double d)
- class **String**:
  - new **String**(char[] chars, int offset, int length)
  - char **charAt**(int index)
  - int **length**()
  - boolean **equals**(String other) / boolean **equalsIgnoreCase**(String other)
  - int **compareTo**(String other)
  - int **indexOf**(char c) / int **indexOf**(String s)
  - String **substring**(int p1, int p2) / String **substring**(int p1)
  - String **concat**(String s) / o usar **+** para concatenar Strings
  - static String **valueOf**(int n)
  - static String **valueOf**(double d)
- class **StringTokenizer**:
  - new **StringTokenizer**(String str)
  - new **StringTokenizer**(String str, String delims)
  - new **StringTokenizer**(String str, String delims, boolean returnDelims)
  - boolean **hasMoreTokens**()
  - String **nextToken**()
- class **Character**
  - static boolean **isDigit**(char c)
  - static boolean **isLetter**(char c)
  - static boolean **isLowerCase**(char c)
  - static boolean **isUpperCase**(char c)
- class **Math**
  - static int **max**(int n1, int n2) / static int **min**(int n1, int n2)
  - static double **max**(double d1, double d2)
  - static double **min**(double d1, double d2)
- char c = **(char)** n; // para int n