Problema 1 (7 puntos)

Apartado a: la clase `Duration` (3 puntos)

Diseña e implementa la siguiente clase para representar duraciones de tiempo en horas, minutos y segundos.

Esta clase deberá tener en cuenta que cada 60 segundos constituyen un minuto más y cada 60 minutos, una hora más. Esto quiere decir que una duración de 3h59m185s, es una duración de 4h2m5s, ya que 185s=3m5s; 59m+3m=62m=1h2m, 3h+1h=4h.

Esta clase tendrá los métodos públicos:

- `new Duration(int hours, int minutes, int seconds)`
  - **tenemos garantizado que los parámetros son no negativos**
  - crea la duración correspondiente a los valores dados.
- `int getHours()`
  - devuelve el número de horas en la duración
  - en el ejemplo dado devolvería 4
- `int getMinutes()`
  - devuelve el número de minutos en la duración
  - en el ejemplo dado devolvería 2
- `int getSeconds()`
  - devuelve el número de segundos en la duración
  - en el ejemplo dado devolvería 5
- `int inSeconds()`
  - devuelve la duración total (contando horas, minutos y segundos) expresada en segundos (en el ejemplo dado 3600*4 + 60 * 2 + 5 = 14525)
- `void add(Duration other)`
  - si `other` es `null` no hace nada
  - añade la duración `other` a la del objeto receptor
- `static Duration add(Duration d1, Duration d2)`
  - suma las dos duraciones y la devuelve como resultado
  - si cualquiera de los parámetros es `null`, devuelve `null`
  - **ninguna de las duraciones pasada como parámetro se modifica**

Apartado b: la clase `DurationAccumulator` (4 puntos)

Se desea tener una colección ampliable de duraciones para poder ir acumulando los tiempos asociados a un determinado identificador.

Como a priori no sabemos el número total de tiempos que queremos controlar, haremos que la clase se adapte a los valores dados.

- `new DurationAccumulator(int initialSize, int marginSize)`

    - **tenemos garantizado** `initialSize` **y** `marginSize` **son positivos**
    - crea espacio para guardar inicialmente `initialSize` duraciones
    - `marginSize` será el margen extra de posiciones que consideraremos al ampliar el tamaño (leer la descripción de `storeDuration`)
- `boolean storeDuration(int id, Duration duration)`
    - si `id` es negativo o `duration` es `null` devuelve falso y no hace nada más
    - si ya existe una duración asociada al `id`, se le añade (suma) la duración dada y se devuelve cierto
    - si no existe se añade en la colección en la posición `id`; en caso de necesitar más espacio, el nuevo tamaño del array será `id + marginSize`; y se devuelve cierto
- `Duration durationAt(int id)`
    - devuelve la duración correspondiente al identificador dado o `null` en caso de que éste no exista
- `int getMinimum()`
    - devuelve el `id` correspondiente a la duración menor (o `-1` en caso de que no haya ninguna)
    - en caso de haber varias con la duración mínima podéis devolver cualquiera de ellas
    - PISTA: recordad que no tenemos duraciones negativas.

Usad funciones auxiliares (privadas) para no duplicar código o para hacer que se entienda mejor.

Problema 2 (3 puntos)

El programa principal que debéis diseñar ha de:

- crear un acumulador de duraciones con tamaño inicial 5 y margen de 3.
- pedir al usuario una línea con las duraciones separadas por espacios
    - cada duración vendrá expresada como una cuarteta de la forma: `id`**`#`**`horas`**`#`**`minutos`**`#`**`segundos`
        - `id:` identificador (`int`)
        - `horas:` número de horas (`int`)
        - `minutos:` número de minutos (`int`)
        - `segundos:` número de segundos (`int`)
    - **tenéis garantizado que están bien formadas y que tienen los tipos adecuados (aunque podría pasar que alguno de los valores fueran negativos)**
    - **si alguno de los valores es negativo simplemente se ignora y ya no se trata la cuarteta**
- acumular las duraciones dadas (sin tener en cuenta las que eran incorrectas)
- mostrar el identificador asociado a la duración total mínima (o que ésta no existe).

Usad funciones auxiliares en las que descomponer vuestra solución.

Ejemplo de uso:

```
Introduzca las duraciones: 1#1#1#23 2#5#13#44 3#3#12#15 4#1#3#22 5#0#-2#33
La duración mínima se corresponde con el identificador 1.
```

- 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 **Double**
  - static double **parseDouble**(String str)
- class **Integer**:
  - static int **parseInt**(String str)
- class **String**:
  - 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
  - String **trim**()
  - 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 **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)