

	h) Se han creado y utilizado métodos estáticos.
RA5	a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.
RA6	a) Se han escrito programas que utilicen arrays

**EXAMEN:**

**Ejercicio 1: Pluviómetro**

**Se evaluarán los siguientes Resultados de Aprendizaje mediante los Criterios de Evaluación especificados: RA1 CEd, RA1 CEe, RA1 CEg, RA1 CEh, RA3 CEa, RA3 CEb, RA3 CEe, RA2 CEe, RA2 CEg, RA2 CEh, RA5 CEa**

Realizar un programa en Java para introducir las lluvias caídas por horas y minutos, presentando el total de lluvias al final.

El programa debe ir solicitando una cadena con el formato horas-minutos (Por ejemplo: 23-12). Si el formato es correcto, debe separar las horas y los minutos, y obtener un objeto de tipo LocalTime. Para saber si el formato es correcto, no es necesario utilizar expresiones regulares, se puede buscar si simplemente la cadena leída contiene “-” y no es vacía. Es decir, NO hay que validar si las horas y minutos están en los rangos correctos, esto se controlará a través de excepciones.

Una vez encontrado el separador “guion” (horas-minutos) la primera parte corresponderá a las horas y el resto corresponderá a los minutos. Para almacenar las horas y los minutos debéis utilizar el tipo de datos más apropiado.

Cuando tengamos ese objeto LocalTime, se solicitará al usuario las lluvias caídas. Este número debe ser un número real y se irá sumando en un acumulador.

El programa debe controlar al menos las siguientes excepciones:

- **NumberFormatException**, en caso de introducir en la cadena de hora-minuto un valor no numérico o no adecuado.
- **DateTimeException**, en caso de que las horas y minutos no estén en los rangos permitidos para las horas (0-23) o los minutos (0-59).
- **InputMismatchException**, en caso de producirse un error en la entrada de datos numérica (bien sea por número decimal o por número entero).

El programa finaliza cuando se introduce una cadena vacía como horas y minutos. En ese momento, se debe presentar la cadena resultado en la que se ha ido almacenando cada uno de los valores registrados. Puedes utilizar la clase StringBuilder para ir generando la cadena de salida resultado.

En la cadena resultado se debe almacenar la cantidad de lluvia con dos decimales utilizando para ello el método de la clase String que permite dar formato.

**EJEMPLO de EJECUCIÓN 1: Uso de valores válidos en cada caso**

```

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)
15-35

Introduce la lluvia caída para las 15:35:00
34,34

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)
1-2

Introduce la lluvia caída para las 01:02:00
45,5343
  
```



Junta de Andalucía

## EXÁMENES PRESENCIALES FP A DISTANCIA FEBRERO 2024

CICLO FORMATIVO:  
MÓDULO PROFESIONAL:  
FECHA Y HORA:  
DURACIÓN:

DAM Parcial Diferenciada  
Programación  
07/02/2024 de 15:30 a 18:15  
2 horas y 45 minutos

**FPA** Formación Profesional Andaluza

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)

Las lluvias totales por horas son:  
15:35 cantidad: 34.34 litros  
01:02 cantidad: 45.5343 litros  
Total: 79,87 litros

### EJEMPLO de EJECUCIÓN 2: Control de errores por valores incorrectos en la hora de medida

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)  
dnfdsnfnds

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)  
23-sdsdsd  
Error en el formato de entrada, el formato debe ser hh-mm

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)  
24-23  
Error en el formato de horas-minutos, la hora debe estar entre 0 y 23 y los minutos entre 0 y 59

### EJEMPLO de EJECUCIÓN 3: Control de errores por valores incorrectos-en cantidad de lluvia

Introduce las horas y minutos en los que se ha realizado la lectura con formato hh-mm (Enter para terminar)  
1-12

Introduce la lluvia caída para las 01:12:00  
ssdsd  
Error en el formato de entrada de datos

### Ejercicio 2: Operaciones con matrices

**Se evaluarán los siguientes Resultados de Aprendizaje mediante los Criterios de Evaluación especificados: RA1 CEd, RA1 CEe, RA1 CEg, RA3 CEa, RA3 CEb, RA3 CEe, RA2 CEe, RA5 CEa, RA6 CEa, RA4 CEe.**

Dadas las dos matrices de enteros positivos A y B de 3x3 que se os proporcionan, escribir un programa que permita obtener una tercera matriz resultado de enteros (también de 3x3) que contenga para cada una de sus posiciones el resultado de realizar las operaciones que se indican a continuación.

Para cada posición (fila, columna):

- Si el valor de la matriz A y el valor de la matriz B para esa posición son PARES, se realizará la SUMA de esos valores y se almacenará en la misma posición de la matriz resultado.
- Si el valor de la matriz A y el valor de la matriz B para esa posición son IMPARES, se realizará la DIVISIÓN del valor A entre el valor B y se almacenará en la misma posición de la matriz resultado.
- Si uno de los valores es PAR y el otro IMPAR, se realizará la RESTA del valor A menos el valor B y se almacenará en la misma posición de la matriz resultado.

Nota: Por ejemplo, si se están comparando los elementos de la posición (2,3) en las matrices A y B, el resultado debe almacenarse en la posición (2,3) de la matriz resultado.

Se debe mostrar el contenido de las matrices iniciales, el contenido de la matriz resultado y la suma total de números contenidos en la matriz resultado. Para ello, a la vez que realizas los cálculos o

comprobaciones, ve componiendo la cadena que contenga cada matriz y que mostrarás al final del programa.

En la salida se deberá mostrar siempre cada valor usando dos dígitos (ej: 5 -> 05).

<b>EJERCICIO .2. OPERACIONES CON MATRICES</b>	
Matriz A:	
01 02 03 04 05 06 07 08 09	
Matriz B:	
00 06 02 01 05 02 06 01 03	
<b>RESULTADO</b>	
Matriz Resultado:	
01 08 01 03 01 08 01 07 03	
Suma de todos valores de la matriz resultado: 33	

### Ejercicio 3: Implementación de la clase Libro

Se evaluarán los siguientes Resultados de Aprendizaje mediante los Criterios de Evaluación especificados: RA1 CE<sub>d</sub>, RA1 CE<sub>e</sub>, RA3 CE<sub>a</sub>, RA3 CE<sub>b</sub>, RA2 CE<sub>b</sub>, RA2 CE<sub>c</sub>, RA2 CE<sub>d</sub>, RA2 CE<sub>e</sub>, RA2 CE<sub>f</sub>, RA5 CE<sub>a</sub>, RA4 CE<sub>a</sub>, RA4 CE<sub>b</sub>, RA4 CE<sub>c</sub>, RA4 CE<sub>d</sub>, RA4 CE<sub>f</sub>, RA4 CE<sub>h</sub>

Implementa una clase llamada *Libro* que modele un libro. La clase debe tener en cuenta los siguientes aspectos:

#### Atributos de la clase:

- **String titulo:** representa el título del libro.
- **String autor:** indica el autor del libro.
- **int numPaginas:** almacena el número de páginas del libro.
- **int añoPublicacion:** guarda el año de publicación del libro.
- **int numLibros:** atributo estático que almacena el número de libros creados.

#### Métodos constructores:

- **Método constructor de 4 parámetros:** Este método permite instanciar un objeto de la clase *Libro* e inicializa todos sus atributos con los valores proporcionados como parámetros. Debe lanzar una excepción *IllegalArgumentException* si el título está vacío o si el número de páginas o el año de publicación son valores negativos.
- **Método constructor de 3 parámetros:** Este método permite instanciar un objeto de la clase *Libro* sin indicar su autor (se colocará el valor DESCONOCIDO). Debe estar basado en el constructor de 4 parámetros.

#### Métodos de la clase:

- **boolean actualizarNumPaginas(int nuevasPaginas):** Si el número de páginas que se recibe como parámetro no es negativo, se considera la actualización y se devuelve true. En caso contrario, se devuelve false.

- **String obtenerInformacion():** Devuelve una cadena EN MAYÚSCULA con la información del libro, incluyendo título, autor, número de páginas y año de publicación, separando cada campo por el símbolo almohadilla ("#"). Ejemplo: "EL LAZARILLO DE TORMES#DESCONOCIDO#110#1554"
- **void mostrarInformacion():** Imprime por consola la información del libro utilizando el método obtenerInformacion(), con formato donde cada campo lo muestre en una línea.
- **String tipoLibro():** devolverá una cadena indicando si el libro es "CORTO" (menos de 100 páginas), "MEDIANO" (de 100 a 200 páginas) o "LARGO" (más de 200 páginas).
- **int obtenerNumLibros():** método estático que devuelve un entero con la cantidad de libros creados.

**Método main de prueba de la clase:**

1. Crea instancias de la clase *Libro* con datos erróneos para que salte la excepción y captúrala con una estructura *try-catch*.
2. Crea dos instancias de la clase *Libro* con datos válidos y muestra su información utilizando el método *mostrarInformacion()*.
3. Actualiza el número de páginas de uno de los libros y muestra de nuevo su información.

**EJERCICIO 3. PRUEBAS DE LA CLASE LIBRO**

Error!! El título del libro no puede estar vacío.  
Error!! El número de páginas del libro no puede ser negativo.  
Error!! El año de publicación del libro no puede ser negativo.

Número total de libros:2

Información del Libro 1:  
- Título: EL LAZARILLO DE TORMES  
- Autor: DESCONOCIDO  
- Número de páginas: 100  
- Año de publicación: 1554

Información del Libro 2:  
- Título: MALDITA ROMA  
- Autor: SANTIAGO POSTEGUILLO  
- Número de páginas: 896  
- Año de publicación: 2023

Información actualizada del Libro 1:  
- Título: EL LAZARILLO DE TORMES  
- Autor: DESCONOCIDO  
- Número de páginas: 128  
- Año de publicación: 1554