MÉTODOS DE APOYO

Para la realización de los ejercicios de laboratorio, puede utilizar algunos métodos contenidos en clases que le son proporcionadas dentro de los proyectos. A continuación, describimos los métodos que puede utilizar. En algunos casos, el uso de estos métodos es optativo, es decir, puede resolver los ejercicios sin utilizarlos; sin embargo, su utilización facilita la realización de los ejercicios y hace el código más legible.

En todos los casos se trata de métodos estáticos, por lo que para invocarlos debe poner el nombre de la clase que los contiene, seguido del operador punto (.) y el nombre del método.

Clase **Checkers**:

- void check(String textoRestriccion, Boolean condicion): este método comprueba si el parámetro condición es false, en cuyo caso lanza una excepción de tipo IllegalArgumentException, incluyendo textoRestriccion en la descripción de la excepción. Puede utilizar el método para controlar las restricciones de las propiedades. Tenga en cuenta que en el parámetro condición debe escribir una expresión lógica que exprese la restricción que debe cumplirse.
  Ejemplo:
  - Llamada: Checkers.check("El valor de n debe ser mayor de 0", n>0)

Clase **Ficheros**:

- List<String> procesaLineaCSV(String linea): este método divide una línea de texto usando la coma (,) como separador, devolviendo los distintos campos en una lista. El método además se encarga de eliminar los espacios que puedan existir antes o después de cada campo, además de transformar los campos "null" a valores null. Puede utilizar este método cuando implemente un constructor de String, como sustituto del método split() de la clase String.
  Ejemplo:
  - Llamada: Ficheros.procesaLinea("Antonio, Jiménez, null")
  - Salida: una lista de String con estos elementos: ["Antonio", "Jiménez", null]

- List<String> procesaLineaCSV(String línea, String sep): igual que el método anterior, pero permite escoger un separador para los campos distinto a la coma.

- List<String> leeFichero(String path): devuelve una lista de String que contiene cada una de las líneas de texto del fichero cuya ruta y nombre se pasan en el parámetro path.
  Ejemplo:
  - Llamada: Ficheros.leeFichero("datos/personas.csv")
  - Salida: una lista de String con las líneas de texto del fichero.

- List<T> leeFichero(String path, Function<String, T> deString_a_T): devuelve una lista de objetos de algún tipo, creando estos objetos a partir de las líneas de texto del fichero cuya ruta y nombre se pasan en el parámetro path. Para indicarle al método con qué clase deben crearse los objetos, en el segundo parámetro debemos escribir una referencia a un constructor. Para que pueda utilizarse, la clase de la que se crearán los objetos debe disponer de un constructor a partir de String.
  Ejemplo:
  - Llamada: Ficheros.leeFichero("datos/personas.csv", PersonaImpl::new)
  - Salida: una lista de Persona con los objetos creados a partir de las líneas del fichero.

Clase **Imágenes**:

- void show(String titulo, String url): muestra una ventana con el título indicado en la que se dibuja la imagen cuya URL se recibe como parámetro.
  Ejemplo:
  - Llamada: Imagenes.show("Fundamentos de Programación", "https://pbs.twimg.com/profile_images/1135125297/LogoFP_color_72_400x400.png")

Clase **Musica**:

- void reproduceMP3(String uri): reproduce un MP3 cuya URL se pasa como parámetro.
  Ejemplo:
  - Llamada: Musica.reproduceMP3("https://www.soundhelix.com/examples/mp3/SoundHelix-Song-1.mp3")

- void reproduceMP3(String uri, Integer maxSegundos): reproduce maxSegundos segundos de un MP3 cuya URL se pasa como parámetro. Si el MP3 es más corto de maxSegundos segundos, lo reproduce entero.