Proyecto 1 UT5 (para hacer en casa y entregar en GitHub)

Objetivos

Saber:

- *declarar y crear un array unidimensional de un tipo primitivo*
- *manejar array como atributos*
- *recorrer un array unidimensional parcialmente completo*
- *utilizar métodos de la clase Arrays (uso de la API de Java)*
- *declarar, crear y recorrer un array bidimensional de un tipo primitivo*
- *manejar arrays como parámetros y/o como valores de retorno*

Comprender:

- *cómo integrar una clase propia con el resto de una aplicación, en particular, con una interfaz gráfica*

Repasar:

- *sentencias de control : if / while / for*
- *método toString()*
- *métodos estáticos*
- *clase Random*

Antes de empezar

- Este ejercicio es para realizar de forma individual en casa.
- El proyecto de partida está en https://github.com/montsemsanz/ENTRE-01-UT5-Lista. Deberás hacer un *fork* a tu cuenta y clonarlo en tu PC **desde línea de comandos** tal y como se explicó
- Ve completando el proyecto desde BlueJ siguiendo el flujo de trabajo habitual con Git desde la CLI (*git add, git commit, git push*)
- Una vez completado haz un *push* del último *commit* a GitHub
- No olvides abrir un ***pull request*** indicando como comentario el texto "*Terminado proyecto 1 UT5 Lista de números*"
- Se valorará en la corrección que el programa esté probado (compila y ejecuta bien) y que esté claramente escrito y organizado (se respetan las reglas de estilo del lenguaje Java, nombres descriptivos, código no duplicado, ...)

- La fecha tope de entrega es el **Martes 21 Diciembre** hasta las **23,30h**.

- Se anulará automáticamente la corrección del ejercicio y se **evaluará con un 0** si se detecta que ha sido copiado o dejado copiar a algún compañero/a
- Se penalizará si no se siguen las normas de entrega del ejercicio
  - ✗ no se ha hecho un *fork* / no se sube vía *commit*
  - ✗ hay algún *commit* posterior a esta fecha de entrega
  - ✗ no se ha abierto un *pull request*
- El profesorado podrá convocar al alumno/a para defender oralmente el proyecto

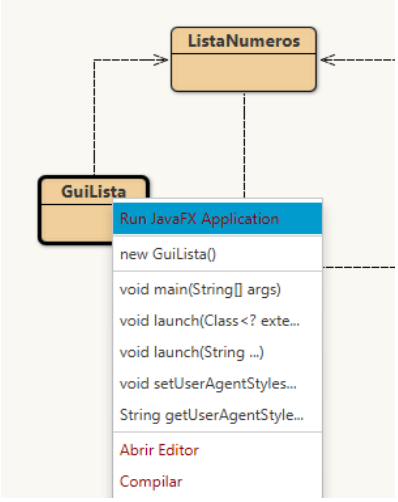
Especificaciones

Haz el *fork* del proyecto **ENTRE-01-UT5-Lista** desde **https://github.com/montsemsanz** a tu cuenta GitHub y clona el proyecto a tu PC.

El proyecto incluye cuatro clases:

- ListaNumeros - modela una lista de nºs enteros. **Es la que tienes que completar y entregar. Pon tu nombre detrás de la etiqueta @author**
- GuiLista - representa el interfaz gráfico para interactuar con el resto de la aplicación. Esta clase está completa y no tienes que modificarla
- TestListaNumeros – contiene código para probar por consola los métodos de la clase ListaNumeros. Esta clase está completa y no tienes que modificarla
- Utilidades - incluye un método estático que habrá que utilizar

Para ver el funcionamiento de la aplicación y saber qué es lo que tienes que conseguir puedes ver este vídeo.



También puedes ejecutar la aplicación (a medida que vayas completando la clase ListaNumeros) ejecutando el interfaz gráfico de la forma: *Botón derecho sobre la clase GuiLista / Run JavaFX Application*

Clase ListaNumeros

El diagrama muestra los atributos y métodos que ha de incluir la clase.

Se dan unas constantes y una variable estática ya definidas.

Completa el resto de la clase ajustándote a la visibilidad, nombres y signaturas que indica el diagrama.

El orden en que has de definir los miembros de la clase es el siguiente:

- dos atributos, una lista de números enteros (representada por un array unidimensional) y el nº actual de elementos guardados en la lista
- **constructor** - crea la lista al tamaño indicado por el parámetro y establece el valor del atributo *pos* a 0.

- **addElemento()** – añade el numero recibido como parámetro siempre al final de la lista modificando adecuadamente el valor de *pos*. Un valor solo se añade a la lista si ésta no está completa. Si se puede añadir devuelve *true*, *false* en otro caso.

- **estaCompleta()** – devuelve *true* si la lista está completa, *false* en otro caso. Hazlo sin *if*

- **estaVacia()** – *devuelve true* si la lista está vacía, *false* en otro caso. Hazlo sin *if*

- **getTotalNumeros() –** accesor que devuelve el nº real de elementos guardados en la lista.

- **vaciarLista()** - vacía la lista (ésta se queda sin elementos)

- **toString()** - devuelve una cadena con la representación textual de la lista. Si la lista está vacía se devuelve "".

  Cada nº en la lista está centrado en tantas posiciones como indica la constante ANCHO_FORMATO. (usa el método centrarNumero() de la clase `Utilidades`)

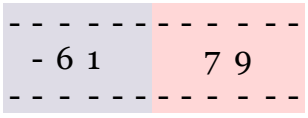
  La lista está enmarcada por dos líneas de cabecera de caracteres CAR_CABECERA.

  Observa que el nº de caracteres en esas líneas depende tanto del tamaño real de la lista como del ancho de formato de cada nº.

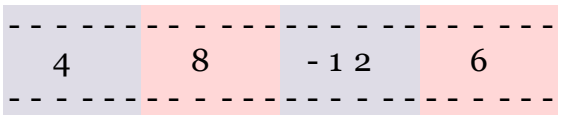
  Evita duplicar código. Puedes crear métodos privados de ayuda para hacer este método.

  Con el ancho de formato y carácter de cabecera que indican las constantes,

  Si *lista* = {-61, 79 } se devuelve como `String`



  Si *lista* = {4, 8, -12, 6} se devuelve como `String`



  Prueba a cambiar los valores de las constantes y comprueba que todo funciona bien.

  Además de ejecutar la aplicación con la parte gráfica testea cada nuevo método que completes con la clase **`TestListaNumeros`**.

- **segundoMaximo()** - calcula y devuelve un entero, el segundo valor máximo en la lista. Si no hay se devuelve *Integer.MIN_VALUE*. No se puede usar ningún otro array auxiliar ni hay que ordenar previamente la lista.

  Si lista = {21, -5, 28, -7, 28, 77, 77, -17, 21, 15, 28, 28, 77} **se devuelve 28**
  lista = {21, -5, 28, -7, 77} **se devuelve 28**
  lista = {77, 21} **se devuelve 21**
  lista = {21} **se devuelve Integer.MIN_VALUE**
  lista = {21, 21, 21, 21} **se devuelve Integer.MIN_VALUE**

- **segundosMaximosAlPrincipio()** - el método coloca los valores que son segundos máximos al principio de la lista respetando el orden de aparición del resto de elementos. No se puede usar ningún otro array auxiliar ni hay que ordenar previamente la lista.

  Se devuelve *true* si se han colocado los segundos máximos, *false* si no se han colocado los segundos máximos porque no había ninguno.

44	28	44	28	13	28	26				*pos = 7*	*Lista inicial*
28	28	28	44	44	13	26				*pos = 7*	*Colocados segundos máximos*
-17	12	44								*pos = 3*	*Lista inicial*
12	-17	44								*pos = 3*	*Colocados segundos máximos*
21	21	21	21							*pos = 4*	*Lista inicial*
21	21	21	21							*pos = 4*	*Colocados segundos máximos*
0	1	2	3	4	5	6	7	8	9		

- **buscarBinario()** - hace una búsqueda binaria de *numero* devolviendo -1 si no se encuentra o la posición en la que aparece si se encuentra (si hay varios iguales se devuelve la posición del primero encontrado). El array original *lista* no se modifica. Para ello crea previamente una copia de *lista* y trabaja con la copia. Usa exclusivamente métodos de la clase `Arrays`. Recuerda el **requisito previo** para hacer una búsqueda binaria en un array.

  Si *lista* = {77, 21, 22 , 15, 21 , 77, 21, -7, 6, -5, 21} y *numero* = *21* devuelve 4

  Si *lista* = {77, 21, 22 , 15, 21 , 77, 21, -7, 6, -5, 21} y *numero* = *87* devuelve -1

- **crearBrillos()** - es un método estático que no tiene parámetros y devuelve un array de 2 dimensiones, *brillos*, con tantas filas y columnas como indique la constante DIMENSION y rellenado con valores aleatorios en el intervalo [0, 10] (inclusive). Estos valores van a representar el brillo de una zona del espacio.

- **detectarEstrellas()** - es un método estático que toma como parámetro el array *brillos* que devuelve el método anterior y construye y devuelve un array de valores *boolean* de las mismas dimensiones con valores *true* en las posiciones donde hay estrellas.

  Una posición *f,c* del array *brillos* es una estrella si la suma del valor de los brillos de sus cuatro vecinos (arriba, abajo, derecha e izquierda) es **mayor que 30**.

  Nota - No hay estrellas en los bordes del array *brillos*.

En los bordes no hay estrellas

1	**6**	**7**	**4**	**8**	**5**	**2**
1	**1**	**3**	**10**	**9**	**2**	**2**
10	**2**	**1**	**4**	**10**	**10**	**10**
9	**3**	**3**	**1**	**2**	**10**	**10**
8	**4**	**6**	**4**	**1**	**10**	**10**
10	**5**	**10**	**10**	**5**	**1**	**7**
2	**7**	**9**	**5**	**4**	**6**	**1**

En los bordes no hay estrellas

10	**4**	**10**	**10**	**2**
10	**8**	**10**	**10**	**4**
6	**9**	**8**	**9**	**9**
10	**7**	**7**	**0**	**6**
10	**4**	**6**	**10**	**1**

Rúbrica evaluación	
atributos	2
construtor	2
add	6
estaCompleta	2
estaVacia	2
getTotalNumeros	2
vaciarLista	2
toString	14
segundoMaximo	14
segundosMaximosAlPrincipio	24
buscarBinario	8
crearBrillos	10
detectarEstrellas	12
	100
Penalización (no compila)	-0,75 (sobre 10)
Penalización (no probado, no se integra con la Gui)	hasta -0,5 (sobre 10)
Penalización (mal estilo: duplica código, otros, ...)	hasta -0,5 (sobre 10)