Recuperatorio segundo examen parcial - AED2, Java 2sem, 2022

Para el segundo examen de la materia entregaremos **tres** clases. Cada archivo **deberá contener en comentario tu nombre completo y número de documento al principio del archivo**.

Los archivos subidos **deberán** llamarse **Operaciones.java, Decodificador.java** y **Display.java**. Dos de ellos se entregan como *esqueleto* para completar. No subir el archivo .txt adjunto a este examen.

Se tomará en cuenta la comprensión de las consignas en cuanto a los requerimientos específicos que se solicitan. No se corregirá un código que no pueda ejecutarse, **no se admiten errores de sintaxis.**

Ejercicio 1:

Abrir la clase Operaciones,java o copiar el código en tu IDE y completar donde se indica. Los requerimientos de cada método se encuentran en comentario antes de la declaración de cada método. Modificar solamente donde se pide.

Ejercicio 2:

Para el ejercicio 2 programaremos una clase llamada “Decodificador”, es decir, entregarán un archivo llamado Decodificador.java. Esta clase se probará utilizando el archivo “coded.txt” que se entrega adjunto a este examen. Nuestra clase deberá cumplir los siguientes requerimientos de diseño:

a) Deberá contener un campo **privado** de tipo entero (int) llamado dni, que tendrá como valor inicial **tu número de documento** (private int dni = xxxxxxxx). Programar un getter para este campo.
b) Programaremos un solo constructor por defecto para esta clase, sin parámetros y sin código dentro.
c) Programaremos una clase pública llamada **leerArchivo**, que tomará un String de parámetro y devolverá una lista de Strings (List<String>) conteniendo todas las líneas del archivo ubicado en la ruta que se pasó como parámetro.
d) Para el método anterior: programar lo necesario para que, si el archivo no existe, se devuelve una lista con un solo elemento: una String “**error**”. Remover cualquier cláusula **throws** que se haya puesto, resolver cualquier excepción que surja en un bloque catch dentro del método.
e) Programaremos un método público llamado **decodificar** que tomará una String de parámetro y devolverá (retornará) un String. Este método tomará cada caracter en el String en parámetro y al número que le corresponda, si lo entendemos como un número (char o int), le restará 2. Retornará la String resultante de ejecutar este procedimiento. Ayuda: se puede utilizar .toCharArray(), (char) (un número), String s = new String(un array de chars); como herramientas para resolver esto, aunque se puede resolver de otras formas.

Ejercicio 3:

Para el ejercicio 3 programaremos una clase llamada “Display”, que se entrega como esqueleto. Abrir esta clase y completarla donde corresponda según las indicaciones.

Mientras programes puedes utilizar prints, herramientas de debug, y cualquier cosa que consideres necesaria para corroborar el buen funcionamiento de tu programa: comentalos antes de entregar. No se corregirá código con errores de sintaxis.

Ejemplos de testing:

Ejercicio 1:

Considerar Operaciones op = new Operaciones();

Test	**Retorno esperado**

`Map<String, String> mapa = new HashMap<>();`
`mapa.put("dummyKey", "dummyValue");`
`op.sizeAndDelete(mapa);`
`mapa.size()`
	1

`Map<String, String> mapa = new HashMap<>();`
`for (int i = 0; i < 70; i++) {`
`  mapa.put("key" + i, "dummyValue");`
`}`
`op.sizeAndDelete(mapa);`
`mapa.size()`
	0

`Map<String, String> mapa = new LinkedHashMap<>();`
`mapa.put("k1", "v1");`
`mapa.put("k2", "v2");`
`List<String> lista = op.valores(mapa);`
`lista.get(0)`
	v1

`lista.get(1)`	v2

`Map<Double, String> mapa = new HashMap<>();`
`mapa.put(3.14, “v1”);`
`mapa.put(3, “v2”);`
`op.sumarLlaves(mapa);`
	6.14

`// En este test se verificará que el archivo en ruta`
`// contenga tu DNI`
`op.writeDNI("hw.txt")`	

Ejercicio 2:

Considerar: Decodificador deco = new Decodificador();

Test	**Retorno esperado**
deco.getDni()	Tu DNI
List<String> codificadas = deco.leerArchivo("coded.txt"); codificadas.size()	100
List<String> codificadas = deco.leerArchivo("no_existe.txt"); codificadas.get0)	**error**
List<String> codificadas = deco.leerArchivo("coded.txt"); codificadas.get(0) 2aVjg"itqwr"swkemn{"wpfgtuvqqf"vjcv"vqzke"ycuvg"ycu"vjg"oquv"ghhgevkxg"dcttkgt"vq"wug"cickpuv"vjg"|qodkgu0 codificadas.get(50) 72aCu"jg"yckvgf"hqt"vjg"ujqygt"vq"ycto."jg"pqvkegf"vjcv"jg"eqwnf"jgct"ycvgt"ejcpig"vgorgtcvwtg0	

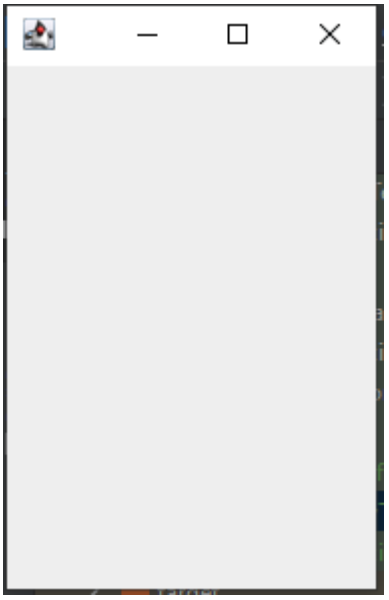
List<String> codificadas = deco.leerArchivo("coded.txt");
deco.decodificar(codificadas.get(6));
 	
6_Flash photography is best used in full sunlight.
 	
deco.decodificar(codificadas.get(99));
 	
99_She was the type of girl who wanted to live in a pink house.	

Ejercicio 3:

considerar que se ejecuta lo siguiente desde un public static void main.

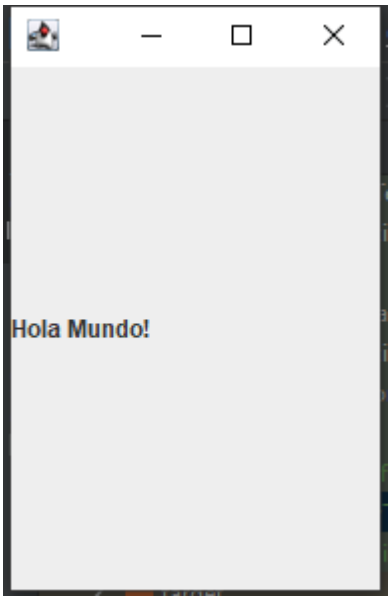
Display display = new Display(200, 300);

Se espera ver una ventana como la siguiente:



```
Display display = new Display(200, 300);
display.mostrar("Hola Mundo");
```

Se espera ver una ventana como la siguiente:



El resultado de ejecutar el main de Display se ve algo así, asumiendo que las últimas cifras del dni son 12. Si estas cifras cambian cambiará solamente el mensaje.