

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



FUNDAMENTOS DE LA PROGRAMACIÓN II - INFORME DE LABORATORIO

Estudiantes:

- Max Junior Soncco Mamani
- Sara Sofia Quispe Diaz
- Lucciano Valentino Tijero Davila

Profesor: Richart Smith Escobedo Quispe

Grupo: "B"

Fecha de Entrega: 24/01/2024

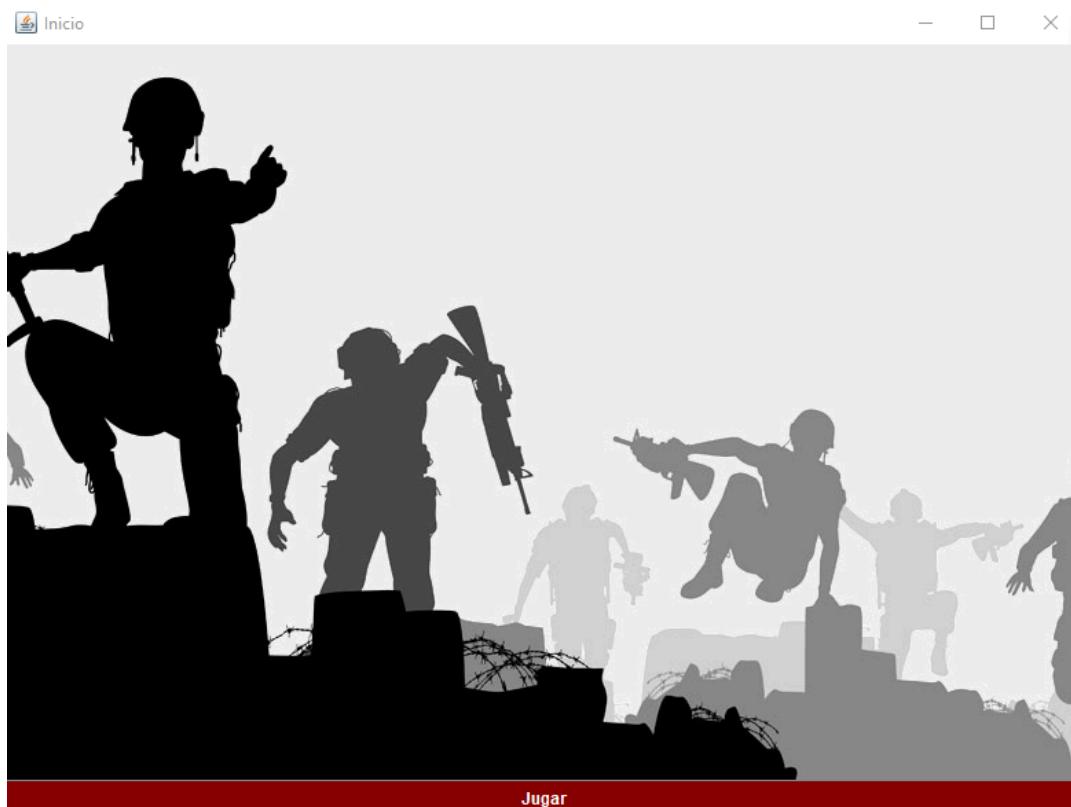
AREQUIPA – PERÚ

2024

Genere un tablero(10x10) para el juego que está desarrollando en laboratorio usando Swing

1. La interfaz generada debe tener las recomendaciones del libro de Jhonson.

Al iniciar el archivo Videojuego.java, tenemos la siguiente ventana:



Al hacer clic en el botón “Jugar”, nos mostrará el tablero creado:

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4			2L5							
5					1C8					
6	1A10							1C10		
7										
8	2E0									
9									1L4	
10										

Dicho tablero ya contiene los elementos necesarios en primera instancia, los soldados dispersos de manera aleatoria por todo el tablero.

2. Averiguar en Java cuanto de memoria RAM utilizó para generar su tablero. (Print en consola)

Al momento de de la prueba, se nos presenta la cuestión del uso de cantidad de RAM del “videojuego”. Dándonos los siguientes resultados en una muestra total de 2 interacciones.

Prueba 1.

```
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/powershell

PS C:\Users\sofiasun\Desktop\TableroGUIv1> & 'C:\Program Files\Java\jdk-21.0.1\bin\java.exe' -cp 'C:\Program Files\Java\jdk-21.0.1\bin\java\DetailsInExceptionMessages' '-cp' 'C:\Users\sofiasun\AppData\Roaming\Code\User\workspace3\redhat.java\jdt_ws\TableroGUIv1_575db2e4\bin' 'Videojuego'
Memoria utilizada por el programa: 6292376 bytes
```

Prueba 2.

```
PS C:\Users\sofiasun\Desktop\TableroGUIv1> & 'C:\Program Files\Java\jdk-21.0.1\bin\java.exe' -cp 'C:\Program Files\Java\jdk-21.0.1\bin\java\DetailsInExceptionMessages' '-cp' 'C:\Users\sofiasun\AppData\Roaming\Code\User\workspace3\redhat.java\jdt_ws\TableroGUIv1_575db2e4\bin' 'Videojuego'
Memoria utilizada por el programa: 7340952 bytes
```

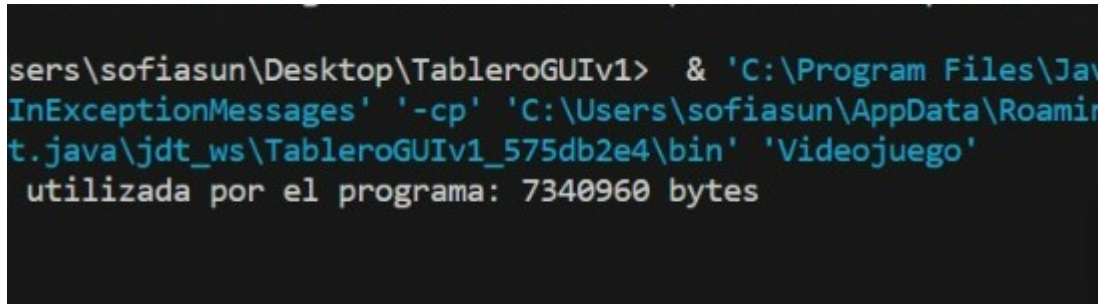
Prueba 3.

```
PS C:\Users\sofiasun\Desktop\TableroGUIv1> & 'C:\Program Files\Java\jdk-21.0.1\bin\java.exe' -cp 'C:\Program Files\Java\jdk-21.0.1\bin\java\DetailsInExceptionMessages' '-cp' 'C:\Users\sofiasun\AppData\Roaming\Code\User\workspace3\redhat.java\jdt_ws\TableroGUIv1_575db2e4\bin' 'Videojuego'
Memoria utilizada por el programa: 7340944 bytes
```

Prueba 4.

```
PS C:\Users\sofiasun\Desktop\TableroGUIv1> & 'C:\Program Files\Java\jdk-21.0.1\bin\java.exe' -cp 'C:\Program Files\Java\jdk-21.0.1\bin\java\DetailsInExceptionMessages' '-cp' 'C:\Users\sofiasun\AppData\Roaming\Code\User\workspace3\redhat.java\jdt_ws\TableroGUIv1_575db2e4\bin' 'Videojuego'
Memoria utilizada por el programa: 7340928 bytes
```

Prueba 5.



```
sers\sofiasun\Desktop\TableroGUIv1> & 'C:\Program Files\Java\jdk-11.0.2\bin\java.exe -Xmx512M -Djava.class.path=C:\Users\sofiasun\AppData\Roaming\Microsoft\Windows\CurrentVersion\Local Settings\Application Data\Java\jdt_ws\TableroGUIv1_575db2e4\bin\Videojuego.jar'
utilizada por el programa: 7340960 bytes
```

Llegamos a una conclusión bastante amplia en la que deducimos que se podría tratar de que la aplicación esté usando más RAM de lo esperado, inferimos que probablemente mantener archivos externos abiertos podría interferir en la cantidad de memoria RAM utilizada.

Ya que:

Prueba 1: 6.00087738037109375mb \approx 6

Prueba 2: 7.00087738037109375mb \approx 7

Prueba 3: 7.0008544921875mb \approx 7

Prueba 4: 7.0008544921875mb \approx 7

Prueba 5: 7.000885009765625mb \approx 7

Lo mismo que nos da una media ponderada de

Promedio = $(6+7+7+7+7)/5 = 6.8\text{mb}$

Observamos que tiene una media de 6.8mb, las cuales no consideramos óptimas para un videojuego de tal medida como el realizado.

3. Qué patrón de diseño de software utilizará para reducir la cantidad de memoria utilizada.

Al indagar sobre los diversos patrones de diseño que pueden ser utilizados al momento de programar y que sirven de gran ayuda para la optimización de recursos y demás cosas, nos encontramos con el patrón Flyweight.

Lo que hicimos fue implementar el patrón mostrado, el cuál se puede explicar de forma sencilla. Los elementos o atributos que comparten todos los objetos que se van a instanciar, pueden ser almacenados en una clase diferente de manera que no se vayan a crear atributos para cada tipo de “Soldado” en este caso. Es por eso que, unimos todos los elementos que comparte cada tipo de soldado y los llevamos a una clase nueva, la cual va a ser inmutable, de esta manera nos aseguramos que sus elementos no cambiarán y podrán ser usados por todos los objetos que vayan a llamar a la clase.

Esto se ve reflejado en la creación de las nuevas clases que llevan por nombre, el nombre de la clase de la que se recogieron los atributos repetitivos más la terminación Flyweight:

```
Arquero.java      Mapa.java
ArqueroFlyweight.java  Proyectiles.java
Caballero.java    Soldado.java
CaballeroFlyweight.java SoldadoConcretoFlyweight.java
Ejercito.java     SoldadoFlyweight.java
Espadachin.java  StartScreenExample.java
EspadachinFlyweight.java TableroGUI.java
Lancero.java      Videojuego.java
LanceroFlyweight.java videojuego.png
```

También se crea la clase SoldadoConcretoFlyweight, la cual va a contener los atributos que pueden usar todos los tipos de soldados en el mapa. Dentro de ellos encontramos nombre, posición, estado y muchos otros.

Después de implementar este patrón, los resultados de la obtención de cantidad de uso de memoria RAM, son los siguientes:

Prueba 1:

```
Memoria utilizada por el programa: 1532000 bytes
Ejercito 1:
Ejercito 2:
```

Prueba 2:

```
Memoria utilizada por el programa: 1612728 bytes
Ejercito 1:
Ejercito 2:
:
```

Prueba 3:

```
Memoria utilizada por el programa: 1530936 bytes
Ejercito 1:
Ejercito 2:
```

Prueba 4:

```
Memoria utilizada por el programa: 1531968 bytes
Ejercito 1:
Ejercito 2:
:
```

Prueba 5:

```
Memoria utilizada por el programa: 1532016 bytes
Ejercito 1:
Ejercito 2:
:
```

Se calcula, nuevamente, la media de los datos presentados, para hacer la interpretación y generar la conclusión.

MB utilizados:

Prueba 1: 1.461029052734375mb \approx 1.46mb

Prueba 2: 1.53801727294921875mb \approx 1.54mb

Prueba 3: 1.46001434326171875mb \approx 1.46mb

Prueba 4: 1.46099853515625mb \approx 1.46mb

Prueba 5: 1.4610443115234375mb \approx 1.46mb

Y se obtiene el promedio de la muestra.

Promedio = $(1.46+1.54+1.46+1.46+1.46)/5=1.476\text{mb}$

CONCLUSIONES:

Observamos que la diferencia de promedios de ambas muestras es abismal, y se tiene confianza en que la implementación del patrón de diseño para la reducción del uso de memoria RAM es de ayuda significativa. Demuestra que es necesario el uso de las herramientas previstas y propuestas que se nos proporciona, ya que son de vital importancia para futuros proyectos y de una magnitud mucho más amplia.

Referencias.

<https://refactoring.guru/es/design-patterns/flyweight/java/example#lang-features>