UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

ASIGNATURA: FUNDAMENTOS DE PROGRAMACIÓN 2

DOCENTE: REVILLA ARROYO CHRISTIAN ALAIN

"Laboratorio 08: HashMap"

Estudiante:

Torres Chávez Jhair Alejandro

Arequipa – Perú

2023

Laboratorio 08:

Enunciado

Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogeneradoaleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldadoscreados y sus puntos de vida (usar caracteres como | _ y otros y distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo.

Link GitHub (CÓDIGO):

Método de resolución:

Primero se creó la clase Soldado.java que como solicitaba en el enunciado, debe contener sus atributos: Nombre, vida, fila y columna, con sus respectivos métodos getters, setters y constructores.

Posteriormente dentro de la clase Videojuego.java se empleó una ArrayList bidimensional para simular el tablero donde se posicionarán los objetos clase Soldado, además se dividirán en 2 ejércitos, cada uno con su propia cantidad de soldados y atributos particulares, para ello usamos arrays estándar para almacenarlos.

Los soldados tendrán atributos aleatorios, excepto el nombre:

Nombre: Autogenerado, de acuerdo a su posición

Vida: [1 - 5]

Fila: [1 - 8]

Columna: [1 - 8]

Los métodos usados se dividen en los de generación (tablero y ejércitos), graficación y auxiliares (mostrar el tablero) y estadísticos (rankeos y clasificación de soldados), Los métodos de generación se encargan de crear y establecer los tamaños de los ejércitos y tener el tablero listo para que se le agreguen los ejércitos. Los métodos de graficaciones y auxiliares, se encargan de comunicar al ArrayList (Tablero) con los 2 HashMap para poder comprar y ubicar a los soldados dentro del tablero y poder graficarlos, de manera distinta el ejército A y ejército B.

Finalmente los métodos estadísticos se encargan de ordenar y clasificar a los soldados de un ejército de acuerdo a su nivel de vida, el ordenamiento usado es el de burbuja, sin modificar Map original, sino retornando uno nuevo.

El ganador del videojuego se basa en la cantidad de soldados que posee cada ejército, quien más posee soldados va a ganar. En caso de igualdad de soldados, es un empate. Además se agregó la iteración por si el usuario desea volver a jugar.

Ejército A:

@@@@@
@@@@@
@@@@@

Ejército B:

$$$$$
$$$$$
$$$$$

Ordenamiento en HashMap

Aunque en el inicio, durante el planteamiento de la lógica fue un problema, posteriormente se pudo modelar y dar solución, ya que cada elemento del HashMap no se identifica por su índice, sino por su Key.

En este caso, para cada Key del HashMap corresponderá a su orden de creación Por ello, tuvimos que usar los siguientes métodos propios del HashMap:

- **ejerAux.putAll(ejer):**

  Este método nos ayuda a copiar un HashMap por completo de uno a otro, sin modificar el original. Lo que nos ayuda a generar y manejar otra estructura y de esta forma poder tener un mayor control de los datos generados originalmente y ordenados.

- **ejerAux.get(j):**

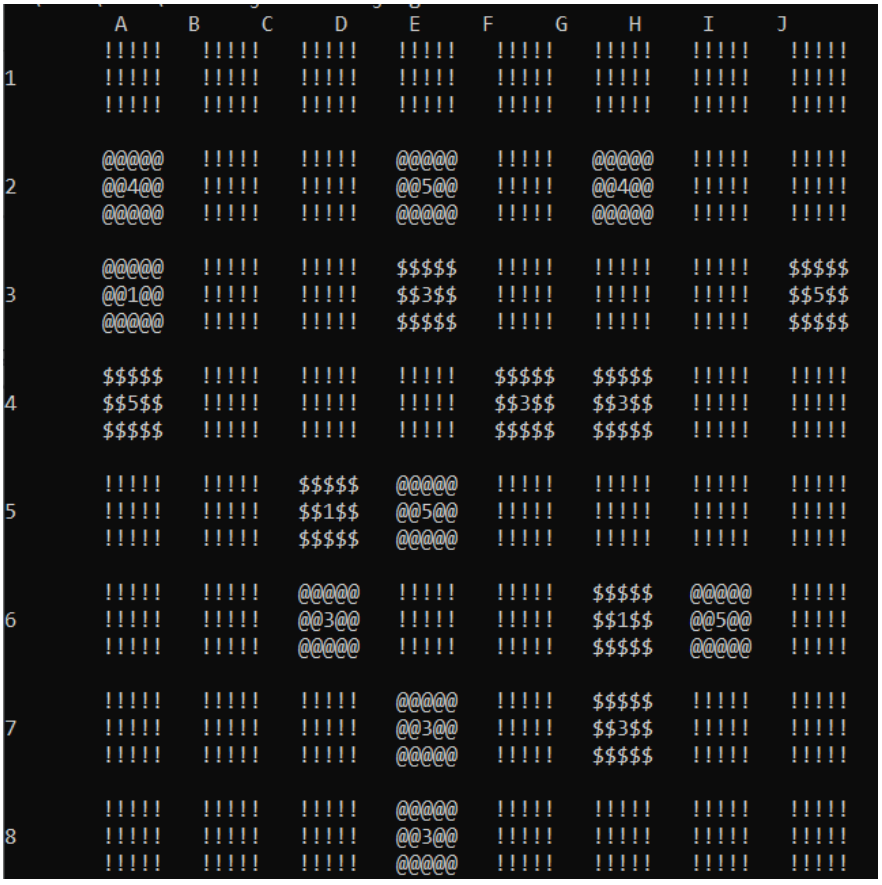
  Este método lo usamos junto a bucles for para poder acceder a cada objeto Soldado dentro del HashMap, ya que cada objeto tiene su propio Key (identificador) según su orden de creación.

- **ejerAux.replace(j , aux):**

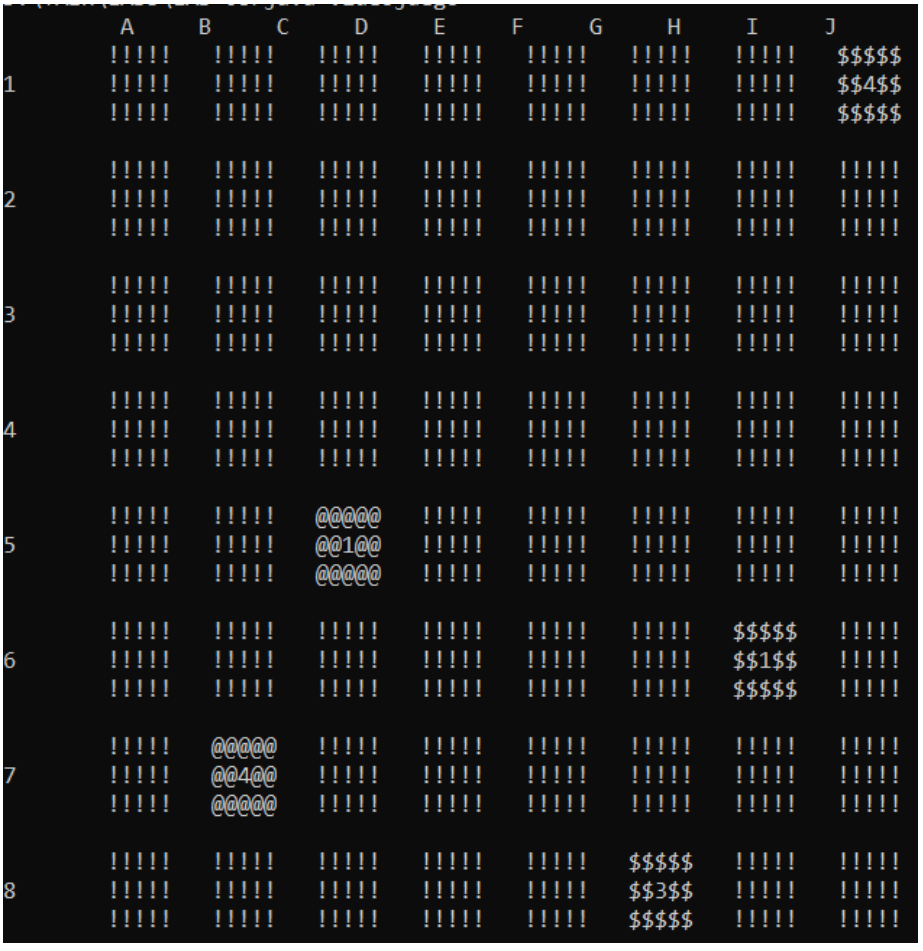
  Este método es usado para poder intercambiar los Values (objeto Soldado), y colocarlos en la Key que queramos (j). Bastante útil en los algoritmos de ordenamiento, de esta manera podremos alterar el orden de los objetos (descendente) según su vida, pero manteniendo las Keys, ya que funciona como su posición dentro de los “Ranking”.

Resultado final

Graficación del Tablero (CASO 1):



Graficación Tablero (CASO 2):



Estadísticas

```
===Estadisticas: EjercitoA===

Soldado TOP vida:
Nombre: Soldado6x1
Vida: 4
Fila : 7 Columa: 2

Promedio vida:
2.5
Orden de creacion:

Posicion 1:
Nombre: Soldado6x1
Vida: 4
Fila : 7 Columa: 2

Posicion 2:
Nombre: Soldado4x2
Vida: 1
Fila : 5 Columa: 3

Ranking de poder:

Posicion 1:
Nombre: Soldado6x1
Vida: 4
Fila : 7 Columa: 2

Posicion 2:
Nombre: Soldado4x2
Vida: 1
Fila : 5 Columa: 3
```

```
===Estadisticas: EjercitoB===

Soldado TOP vida:
Nombre: Soldado0x7
Vida: 4
Fila : 1 Columa: 8

Promedio vida:
2.6666666666666665
Orden de creacion:

Posicion 1:
Nombre: Soldado7x5
Vida: 3
Fila : 8 Columa: 6

Posicion 2:
Nombre: Soldado5x6
Vida: 1
Fila : 6 Columa: 7

Posicion 3:
Nombre: Soldado0x7
Vida: 4
Fila : 1 Columa: 8

Ranking de poder:

Posicion 1:
Nombre: Soldado0x7
Vida: 4
Fila : 1 Columa: 8

Posicion 2:
Nombre: Soldado7x5
Vida: 3
Fila : 8 Columa: 6

Posicion 3:
Nombre: Soldado5x6
Vida: 1
Fila : 6 Columa: 7
Resultado Final:
GANO TEAM B
Continuar? (Y/N)
```