
	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

# INFORME DE LABORATORIO

## (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2				
TÍTULO DE LA PRÁCTICA:	HashMap				
NÚMERO DE PRÁCTICA:	08	AÑO LECTIVO:	2024 B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	29/11/2024	HORA DE PRESENTACIÓN	18:10:00		
INTEGRANTE (s) José Manuel Moroco Saico				NOTA (0-20)	
DOCENTE(s): Ing. Lino Jose Pinto Oppe					



RESULTADOS Y PRUEBAS
<div>I. EJERCICIOS RESUELTOS:</div> <div></div>

### *Clase Soldado:*

```
LABORATORIO_08 > J Soldado.java > ...
1  package LABORATORIO_08;
2
3  // CLASE QUE REPRESENTA A UN SOLDADO EN EL JUEGO O SISTEMA
4  public class Soldado {
5
6      // ATRIBUTOS PRIVADOS PARA ALMACENAR EL NOMBRE, VIDA Y POSICIÓN DEL SOLDADO
7      private String nombre; // NOMBRE DEL SOLDADO
8      private int vida; // PUNTOS DE VIDA DEL SOLDADO
9      private int fila; // POSICIÓN EN LA FILA DEL TABLERO O ESCENARIO
10     private int columna; // POSICIÓN EN LA COLUMNA DEL TABLERO O ESCENARIO
11
12     // CONSTRUCTOR QUE INICIALIZA LOS ATRIBUTOS DEL SOLDADO
13     public Soldado(String nombre, int puntosVida, int fila, int columna) {
14         this.nombre = nombre; // ASIGNA EL NOMBRE AL SOLDADO
15         this.vida = puntosVida; // ASIGNA LOS PUNTOS DE VIDA INICIALES
16         this.fila = fila; // ASIGNA LA FILA DONDE SE UBICA EL SOLDADO
17         this.columna = columna; // ASIGNA LA COLUMNA DONDE SE UBICA EL SOLDADO
18     }
19
20     // MÉTODO PARA OBTENER EL NOMBRE DEL SOLDADO
21     public String getNombre() {
22         return nombre;
23     }
24
25     // MÉTODO PARA OBTENER LOS PUNTOS DE VIDA DEL SOLDADO
26     public int getVida() {
27         return vida;
28     }
29
30     // MÉTODO PARA OBTENER LA FILA DONDE ESTÁ UBICADO EL SOLDADO
31     public int getFila() {
32         return fila;
33     }
34
35     // MÉTODO PARA OBTENER LA COLUMNA DONDE ESTÁ UBICADO EL SOLDADO
36     public int getColumna() {
37         return columna;
38     }
39
40     // MÉTODO QUE DEVUELVE UNA REPRESENTACIÓN EN TEXTO DEL SOLDADO
41     public String toString() {
42         // IMPRIME EL NOMBRE, PUNTOS DE VIDA Y POSICIÓN EN UNA CADENA
43         return nombre + " (Vida: " + vida
44             + " | Pos: [" + (fila + 1) + "," + (columna + 1) + "])";
45     }
46 }
```

## Main:

```
1 package LABORATORIO_08;
2 import java.util.*;
3 public class VideoJuego5 {
4
5     Run | Debug
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         // BIENVENIDA AL USUARIO
10        System.out.println(x:"¡Bienvenido al simulador de batalla!");
11
12        // CICLO PRINCIPAL QUE PERMITE REPETIR EL PROCESO SEGÚN LA ELECCIÓN DEL USUARIO
13        boolean continuar = true;
14        while (continuar) {
15            Random random = new Random();
16            int tTablero = 10; // TAMAÑO DEL TABLERO (10x10)
17
18            // GENERAMOS NÚMERO ALEATORIO DE SOLDADOS PARA CADA EJÉRCITO
19            int nSoldados1 = random.nextInt(bound:10) + 1; // SOLDADOS EJÉRCITO 1
20            int nSoldados2 = random.nextInt(bound:10) + 1; // SOLDADOS EJÉRCITO 2
21            System.out.println("Se generarán " + nSoldados1 + " Soldados para el Ejército1");
22            System.out.println("Se generarán " + nSoldados2 + " Soldados para el Ejército2");
23
24            // CREACIÓN DEL TABLERO COMO ARREGLO BIDIMENSIONAL VACÍO
25            Soldado[][] tablero = new Soldado[tTablero][tTablero];
26
27            // INICIALIZACIÓN DE EJÉRCITOS COMO HASHMAPS
28            HashMap<Integer, Soldado> ejercito1 = new HashMap<>();
29            HashMap<Integer, Soldado> ejercito2 = new HashMap<>();
30
31            // COLOCAMOS LOS SOLDADOS EN EL TABLERO Y LOS ASIGNAMOS A SUS EJÉRCITOS
32            ingresarSoldadosTablero(nSoldados1, tablero, ejercito1, idEjercito:1);
33            ingresarSoldadosTablero(nSoldados2, tablero, ejercito2, idEjercito:2);
34
35            // MOSTRAMOS EL ESTADO DEL TABLERO
36            mostrarTablero(tablero);
37
38            // MOSTRAMOS LAS ESTADÍSTICAS DE CADA EJÉRCITO
39            mostrarEstadisticas(ejercito1, nombreEjercito:"Ejército 1");
40            mostrarEstadisticas(ejercito2, nombreEjercito:"Ejército 2");
41
42            // COMPARAMOS LAS VIDAS TOTALES PARA DETERMINAR EL GANADOR
43            String ganador;
44            if (sumaVida(ejercito1) > sumaVida(ejercito2)) {
45                ganador = "Ejército 1";
46            } else {
47                ganador = "Ejército 2";
48            }
49            System.out.println("El ganador de la batalla es: " + ganador);
50
51            // PREGUNTAMOS SI EL USUARIO QUIERE REALIZAR OTRA SIMULACIÓN
52            System.out.print(s:"¿Desea realizar otra simulación? (s/n): ");
53            String respuesta = scanner.nextLine().toLowerCase();
54            if (!respuesta.equals(anObject:"s")) {
55                continuar = false; // TERMINAMOS EL CICLO SI LA RESPUESTA NO ES "s"
56            }
57        }
58        // DESPEDIDA
59        System.out.println(x:"¡Gracias por jugar! Hasta la próxima.");
60    }
61 }
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 4</p>

### Descripción del Main :

- **Bienvenida y ciclo principal:**

- Se muestra un mensaje inicial y se inicia un ciclo que permite repetir la simulación según la decisión del usuario.

- **Inicialización:**

- Se define un tablero de 10x10, y se generan aleatoriamente la cantidad de soldados para cada ejército usando la clase `Random`.

- **Distribución y visualización:**

- Los soldados se colocan en el tablero y se asignan a cada ejército (`HashMap`). Luego, se muestra el estado del tablero.

- **Cálculo y resultados:**

- Se calculan las estadísticas de los ejércitos y se determina al ganador comparando las vidas totales.

- **Interacción:**

- Se pregunta si desea realizar otra simulación. Si no, se termina el programa con un mensaje de despedida.

### *Métodos usados en el main:*

#### 1. sumaVida

```
61 // ---> MÉTODO PARA SUMAR LA VIDA TOTAL DE UN EJÉRCITO
62 public static int sumaVida(HashMap<Integer, Soldado> ejercito) {
63     int suma = 0;
64     for (Soldado soldado : ejercito.values()) {
65         suma += soldado.getVida(); // SUMAMOS LA VIDA DE CADA SOLDADO
66     }
67     return suma;
68 }
```

Calcula la vida total de un ejército:

- Recorre los valores del `HashMap` del ejército.
- Suma la vida de cada soldado utilizando su método `getVida`.
- Devuelve el total de la suma como un entero.

**Propósito:** Evaluar la fuerza total del ejército con base en la vida acumulada de sus soldados.

#### 2. ingresarSoldadosTablero

```
70 // ---> MÉTODO PARA INGRESAR SOLDADOS EN EL TABLERO Y EJÉRCITO
71 public static void ingresarSoldadosTablero(int nSoldados, Soldado[][] tablero, HashMap<Integer, Soldado> ejercito,
72     int idEjercito) {
73     Random random = new Random();
74     int tTablero = tablero.length; // TAMAÑO DEL TABLERO
75
76     for (int i = 0; i < nSoldados; i++) {
77         int fila, columna;
78         do {
79             fila = random.nextInt(tTablero); // GENERAMOS COORDENADAS ALEATORIAS
80             columna = random.nextInt(tTablero);
81         } while (tablero[fila][columna] != null); // VERIFICAMOS QUE LA CASILLA ESTÉ VACÍA
82
83         int vida = random.nextInt(bound:5) + 1; // ASIGNAMOS UNA VIDA ALEATORIA (1-5)
84         String nombre = "Soldado" + i + "X" + idEjercito;
85         Soldado soldado = new Soldado(nombre, vida, fila, columna);
86         tablero[fila][columna] = soldado; // COLOCAMOS AL SOLDADO EN EL TABLERO
87         ejercito.put(i, soldado); // AÑADIMOS AL SOLDADO AL EJÉRCITO
88     }
89 }
```

Distribuye soldados en el tablero y los registra en el ejército:

- Genera posiciones aleatorias para cada soldado, asegurándose de que no estén ocupadas.
- Asigna atributos como nombre y vida aleatoria (1 a 5) al soldado.

- Coloca al soldado en el tablero y lo añade al `HashMap` del ejército con un identificador único.

**Propósito:** Representar la ubicación y características de los soldados en el tablero y asociarlos con su respectivo ejército.

### 3. mostrarTablero

```

91 // ---> MÉTODO PARA MOSTRAR EL TABLERO
92 public static void mostrarTablero(Soldado[][] tablero) {
93     // MOSTRAMOS LAS LETRAS DE LAS COLUMNAS
94     System.out.print(s:" ");
95     for (char letra = 'A'; letra < 'A' + tablero.length; letra++) {
96         System.out.print(" " + letra + " ");
97     }
98     System.out.println();
99
100    // RECORREMOS FILAS Y COLUMNAS DEL TABLERO
101    for (int i = 0; i < tablero.length; i++) {
102        System.out.printf(format:"%2d ", i + 1); // NÚMERO DE FILA
103        for (int j = 0; j < tablero[i].length; j++) {
104            Soldado soldado = tablero[i][j];
105            if (soldado == null) { // CASILLA VACÍA
106                System.out.print(s:"| ____");
107            } else if (soldado.getNombre().contains(s:"X1")) { // SOLDADO DEL EJÉRCITO 1
108                System.out.print(s:"| S1 ");
109            } else { // SOLDADO DEL EJÉRCITO 2
110                System.out.print(s:"| S2 ");
111            }
112        }
113        System.out.println(x:"|"); // CERRAMOS LA FILA
114    }
115 }

```

Visualiza el estado del tablero:

- Imprime encabezados con las letras de las columnas.
- Muestra cada celda, indicando si está vacía o contiene un soldado del Ejército 1 (S1) o Ejército 2 (S2).

**Propósito:** Facilitar la comprensión visual del estado actual de la batalla.

#### 4. mostrarEstadisticas

```
117 // ---> MÉTODO PARA MOSTRAR ESTADÍSTICAS DE UN EJÉRCITO
118 public static void mostrarEstadisticas(HashMap<Integer, Soldado> ejercito, String nombreEjercito) {
119     System.out.println("\nEstadísticas de " + nombreEjercito + ":");
120
121     Soldado soldadoMayorVida = null; // SOLDADO CON MAYOR VIDA
122     int sumaVida = 0; // SUMA TOTAL DE VIDAS
123
124     // RECORREMOS TODOS LOS SOLDADOS DEL EJÉRCITO
125     for (Soldado soldado : ejercito.values()) {
126         sumaVida += soldado.getVida(); // SUMAMOS LAS VIDAS
127         if (soldadoMayorVida == null || soldado.getVida() > soldadoMayorVida.getVida()) {
128             soldadoMayorVida = soldado; // ACTUALIZAMOS AL SOLDADO CON MÁS VIDA
129         }
130     }
131
132     double promedioVida = (ejercito.size() > 0) ? (double) sumaVida / ejercito.size() : 0; // PROMEDIO DE VIDAS
133     promedioVida = Math.round(promedioVida * 100.0) / 100.0; // REDONDEO A 2 DECIMALES
134
135     // MOSTRAMOS LOS RESULTADOS
136     System.out.println("Soldado con mayor vida: " + soldadoMayorVida);
137     System.out.println("Promedio de vida: " + promedioVida);
138
139     // ORDENAMOS Y MOSTRAMOS SOLDADOS
140     System.out.println(x:"\nRanking de soldados (burbuja):");
141     List<Soldado> soldadosOrdenadosBurbuja = new ArrayList<>(ejercito.values());
142     ordenarPorVidaBurbuja(soldadosOrdenadosBurbuja);
143     for (Soldado soldado : soldadosOrdenadosBurbuja) {
144         System.out.println(soldado);
145     }
146     System.out.println(x:"\nRanking de soldados (selección):");
147     List<Soldado> soldadosOrdenadosSeleccion = new ArrayList<>(ejercito.values());
148     ordenarPorVidaSeleccion(soldadosOrdenadosSeleccion);
149     for (Soldado soldado : soldadosOrdenadosSeleccion) {
150         System.out.println(soldado);
151     }
152 }
153 }
```

Presenta estadísticas detalladas del ejército:

- Determina el soldado con mayor vida y calcula la vida promedio.
- Ordena los soldados en dos rankings (burbuja y selección) y los muestra.

**Propósito:** Proveer información estratégica sobre el desempeño del ejército y comparar soldados por su resistencia.

## 5. ordenarPorVidaBurbuja

```
157 // ---> MÉTODO BURBUJA PARA ORDENAR SOLDADOS POR VIDA
158 public static void ordenarPorVidaBurbuja(List<Soldado> soldados) {
159     for (int i = 0; i < soldados.size() - 1; i++) {
160         for (int j = 0; j < soldados.size() - i - 1; j++) {
161             if (soldados.get(j).getVida() < soldados.get(j + 1).getVida()) {
162                 Soldado temp = soldados.get(j); // INTERCAMBIAMOS POSICIONES
163                 soldados.set(j, soldados.get(j + 1));
164                 soldados.set(j + 1, temp);
165             }
166         }
167     }
168 }
169 }
```

Ordena los soldados por vida de forma descendente usando el método burbuja:

- Compara cada par de soldados adyacentes.
- Intercambia posiciones si el soldado actual tiene menos vida que el siguiente.

**Propósito:** Organizar a los soldados para mostrar sus estadísticas de manera ordenada.

## 6. ordenarPorVidaSeleccion

```
170 // ---> MÉTODO SELECCION PARA ORDENAR SOLDADOS POR VIDA
171 public static void ordenarPorVidaSeleccion(List<Soldado> soldados) {
172     // --> ITERAMOS SOBRE TODOS LOS SOLDADOS (EXCEPTO EL ÚLTIMO, YA QUE QUEDARÁ
173     // ORDENADO AUTOMÁTICAMENTE)
174     for (int i = 0; i < soldados.size() - 1; i++) {
175         // --> ASUMIMOS QUE EL SOLDADO EN LA POSICIÓN ACTUAL TIENE LA VIDA MÁS ALTA
176         int maxIdx = i;
177
178         // --> COMPARAR EL SOLDADO EN maxIdx CON LOS SOLDADOS RESTANTES
179         for (int j = i + 1; j < soldados.size(); j++) {
180             // --> SI ENCONTRAMOS UN SOLDADO CON MAYOR VIDA, ACTUALIZAMOS maxIdx
181             if (soldados.get(j).getVida() > soldados.get(maxIdx).getVida()) {
182                 maxIdx = j; // --> NUEVA POSICIÓN DEL SOLDADO CON MAYOR VIDA
183             }
184         }
185
186         // --> INTERCAMBIAMOS EL SOLDADO EN maxIdx CON EL SOLDADO EN LA POSICIÓN ACTUAL
187         // (i)
188         Soldado temp = soldados.get(maxIdx);
189         soldados.set(maxIdx, soldados.get(i));
190         soldados.set(i, temp);
191     }
192 }
193 }
```

Ordena soldados por vida descendente usando el método selección:



- Encuentra el soldado con mayor vida en la porción no ordenada.
- Lo coloca al inicio de esa porción, iterando hasta que todos estén ordenados.

**Propósito:** Ofrecer una alternativa al método burbuja para ordenar soldados por vida.

## II. PRUEBAS

```

¡Bienvenido al simulador de batalla!
Se generarán 6 Soldados para el Ejército1
Se generarán 1 Soldados para el Ejército2

  A  B  C  D  E  F  G  H  I  J
1  |  |  |  |  |  |  |  |  |  |
2  |  |  |  |  |  |  |  |  |  |
3  |  |  |  |  | S1 |  |  |  |  |
4  |  |  |  |  |  |  |  |  |  |
5  |  |  |  |  |  |  |  |  |  |
8  |  |  |  | S1 |  |  |  |  |  |
9  |  |  |  |  |  |  |  |  |  |
10 |  |  |  |  |  | S1 |  |  |  |

Estadísticas de Ejército 1:
Soldado con mayor vida: Soldado1X1 (Vida: 5 | Pos: [8,4])
Promedio de vida: 3.5

Ranking de soldados (burbuja):
Soldado1X1 (Vida: 5 | Pos: [8,4])
Soldado5X1 (Vida: 5 | Pos: [10,6])
Soldado2X1 (Vida: 4 | Pos: [6,7])
Soldado4X1 (Vida: 3 | Pos: [6,4])
Soldado0X1 (Vida: 2 | Pos: [3,5])
Soldado3X1 (Vida: 2 | Pos: [7,5])

Ranking de soldados (selección):
Soldado1X1 (Vida: 5 | Pos: [8,4])
Soldado5X1 (Vida: 5 | Pos: [10,6])
Soldado2X1 (Vida: 4 | Pos: [6,7])
Soldado4X1 (Vida: 3 | Pos: [6,4])
Soldado3X1 (Vida: 2 | Pos: [7,5])
Soldado0X1 (Vida: 2 | Pos: [3,5])

Estadísticas de Ejército 2:
Soldado con mayor vida: Soldado0X2 (Vida: 3 | Pos: [6,3])
Promedio de vida: 3.0

Ranking de soldados (burbuja):
Soldado0X2 (Vida: 3 | Pos: [6,3])

Ranking de soldados (selección):
Soldado0X2 (Vida: 3 | Pos: [6,3])
El ganador de la batalla es: Ejército 1

```

```

¿Desea realizar otra simulación? (s/n): s
Se generarán 6 Soldados para el Ejército1
Se generarán 8 Soldados para el Ejército2

```

	A	B	C	D	E	F	G	H	I	J
1										
2	S2			S2		S2			S2	
3		S1								
4							S2			
5										
6			S2			S1		S1	S2	
7		S1			S1		S1			
8										
9					S2					
10										

```

Estadísticas de Ejército 1:
Soldado con mayor vida: Soldado1X1 (Vida: 3 | Pos: [7,5])
Promedio de vida: 1.83

Ranking de soldados (burbuja):
Soldado1X1 (Vida: 3 | Pos: [7,5])
Soldado2X1 (Vida: 3 | Pos: [7,2])
Soldado3X1 (Vida: 2 | Pos: [3,2])
Soldado8X1 (Vida: 1 | Pos: [6,8])
Soldado4X1 (Vida: 1 | Pos: [7,7])
Soldado5X1 (Vida: 1 | Pos: [6,6])

Ranking de soldados (selección):
Soldado1X1 (Vida: 3 | Pos: [7,5])
Soldado2X1 (Vida: 3 | Pos: [7,2])
Soldado3X1 (Vida: 2 | Pos: [3,2])
Soldado8X1 (Vida: 1 | Pos: [6,8])
Soldado4X1 (Vida: 1 | Pos: [7,7])
Soldado5X1 (Vida: 1 | Pos: [6,6])

Estadísticas de Ejército 2:
Soldado con mayor vida: Soldado4X2 (Vida: 5 | Pos: [2,9])
Promedio de vida: 2.25

Ranking de soldados (burbuja):
Soldado4X2 (Vida: 5 | Pos: [2,9])
Soldado6X2 (Vida: 3 | Pos: [2,6])
Soldado7X2 (Vida: 3 | Pos: [2,1])
Soldado3X2 (Vida: 2 | Pos: [6,3])
Soldado5X2 (Vida: 2 | Pos: [4,7])
Soldado8X2 (Vida: 1 | Pos: [2,4])
Soldado1X2 (Vida: 1 | Pos: [6,9])
Soldado2X2 (Vida: 1 | Pos: [9,5])

Ranking de soldados (selección):
Soldado4X2 (Vida: 5 | Pos: [2,9])
Soldado6X2 (Vida: 3 | Pos: [2,6])
Soldado7X2 (Vida: 3 | Pos: [2,1])
Soldado3X2 (Vida: 2 | Pos: [6,3])
Soldado5X2 (Vida: 2 | Pos: [4,7])
Soldado8X2 (Vida: 1 | Pos: [2,4])
Soldado1X2 (Vida: 1 | Pos: [6,9])
Soldado2X2 (Vida: 1 | Pos: [9,5])



El ganador de la batalla es: Ejército 2

¿Desea realizar otra simulación? (s/n): n
¡Gracias por jugar! Hasta la próxima.

```

**RUBRICA:**

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
TOTAL		20		19	

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 12</p>

## MIS COMMITS:

```
PS C:\Users\usuario\Desktop\Unsa\RepositorioLocal\FP2> git log
commit 49f7f1d3d88734db5c70e6f7a9eacd8090845cc (HEAD -> main, origin/main, origin/HEAD)
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Nov 29 18:07:17 2024 -0500

    Se corrige un ligero error en los metodos

commit 3cc10090439322e934a9100883ed411aefc23fe6
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Nov 29 17:59:57 2024 -0500

    Se agregan comentarios al main para mejorar la compresion del codigo

commit d721c3f61e4596e9f8a905317b1723861e9690c7
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Nov 29 17:52:47 2024 -0500

    Se modifica solo el main para que el programa sea iterativo

commit 2e55911806775248392ea4e86bebe7caf694d85b
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Nov 29 17:45:50 2024 -0500

    Se agregan comentarios a la clase Soldado para entenes sus atributos y metodos

commit 84dc9efeb5fb1869f7268dd9f7d70887e4840cdd
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Nov 29 01:18:42 2024 -0500

    Se agrega el metodo de ordenamiento por seleccion con HashMap

commit 5578b06e0a605657ce4a3a272357852002d0129b
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Thu Nov 28 22:37:58 2024 -0500

    Se agrega el metodo de ordenamiento Burbuja con HashMap

commit 4957561462ed7ea2ce1cc43e90691cb523152a38
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Thu Nov 28 22:35:22 2024 -0500

    Se agrega el metodo para mostrar las estadisticas del ejercito para que trabaje con HashMap

commit 47b6a1a0dd52ddf877e7cc2a39d1f5e9586dd4c3
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Tue Nov 26 10:10:34 2024 -0500

    Se generan los soldados con Hashmap y se modifica el metodo IngresarSoldados para trabajar con Hashmap



commit 3089aa89acf078504e06237267e7f77f89de9082
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Tue Nov 26 10:10:16 2024 -0500

    Se generan los soldados con Hashmap y se modifica el metodo IngresarSoldados para trabajar con Hashmap

commit 4ea2d375ebe54507e490c5ccd867b1e1a9f51d1c
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Tue Nov 26 09:54:06 2024 -0500

    Se agrega la clase Soldado.java
```

Aca esta el historial de los commits que hice en el laboratorio

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p align="center"><b>Código:</b> GUIA-PRLE-001</p>	<p align="right"><b>Página:</b> 13</p>

En github:



Commits on Nov 29, 2024		
Se corrige un ligero error en los metodos	JoseMorocco committed 18 minutes ago	49f7f1d  <>
Se agregan comentarios al main para mejorar la compresion del codigo	JoseMorocco committed 25 minutes ago	3cc1009  <>
Se modifica solo el main para que el programa sea iterativo	JoseMorocco committed 32 minutes ago	d721c3f  <>
Se agregan comentarios a la clase Soldado para entenes sus atributos y metodos	JoseMorocco committed 39 minutes ago	2e55911  <>
Se agrega el metodo de ordenamiento por seleccion con HashMap	JoseMorocco committed 17 hours ago	84dc9ef  <>
Commits on Nov 28, 2024		
Se agrega el metodo de ordenamiento Burbuja con HashMap	JoseMorocco committed 19 hours ago	5578b06  <>
Se agrega el metodo para mostrar las estadisticas del ejercito para que trabaje con HashMap	JoseMorocco committed 19 hours ago	4957561  <>
Commits on Nov 26, 2024		
Se generan los soldados con Hashmap y se modifica el metodo IngresarSoldados para trabajar con Hashmap	JoseMorocco committed 3 days ago	47b6a1a  <>
Se generan los soldados con Hashmap y se modifica el metodo IngresarSoldados para trabajar con Hashmap	JoseMorocco committed 3 days ago	3089aa8  <>
Se agrega la clase Soldado.java	JoseMorocco committed 3 days ago	4ea2d37  <>
Se agrega la clase Soldado.java	JoseMorocco committed 3 days ago	be29b9f  <>

Link a mi Repositorio:

***<https://github.com/JoseMorocco/FP2>***

## CONCLUSIONES

En este laboratorio, el uso de `HashMap` fue clave para manejar los datos de los soldados de manera organizada, permitiendo asociar fácilmente cada soldado con su identificador. Esto facilitó tanto la asignación inicial como la

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 14

consulta de estadísticas y el manejo de cada ejército. La implementación se enfocó en aprovechar su capacidad para acceder a los datos de forma rápida y ordenada, mejorando la estructura general del simulador de batalla.

### METODOLOGÍA DE TRABAJO

- 1.-Investigar mejor los métodos de búsqueda y ordenamiento
- 2.-Elaborar un pequeño pseudocódigo para plantear el programa
- 3.-Elaborar un diagrama de flujo para ver las opciones que quiero que tenga
- 4.-Implementarlas en el programa
- 5.-Corregir errores

### REFERENCIAS Y BIBLIOGRAFÍA