

Informe de Laboratorio 12

Tema: Clase Soldado - Menú

Nota

Estudiante	Escuela	Asignatura
Ryan Fabian Valdivia Segovia rvaldiviase@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
12	Clase Soldado - Menú	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 06 de Diciembre 2023	Al 11 de Diciembre 2023

1. Tarea

- Puede reutilizar todo el código del laboratorio 11, pero ahora el objetivo es gestionar los ejércitos autogenerados.
- Al ejecutar el videojuego, el programa deberá dar las opciones:

1.1. 1. Juego Rápido

- (Tal cual como en el laboratorio 11) Al acabar el juego mostrar las opciones de volver a jugar y de volver al menú principal. También se deberá tener la posibilidad de cancelar el juego actual en cualquier momento, permitiendo escoger entre empezar un juego totalmente nuevo o salir al menú principal.

1.2. 2. Juego Personalizado

- Permite gestionar ejércitos. Primero se generan los 2 ejércitos con sus respectivos soldados y se muestran sus datos. Luego se tendrá que escoger cuál de los 2 ejércitos se va a gestionar, después se mostrarán las siguientes opciones:
- 1) Crear Soldado: permitirá crear un nuevo soldado personalizado y añadir al final del ejército (recordar que límite es de 10 soldados por ejército)
- 2) Eliminar Soldado (no debe permitir un ejército vacío)

- 3) Clonar Soldado (crea una copia exacta del soldado) y se añade al final del ejército (recordar que límite es de 10 soldados por ejército)
- 4) Modificar Soldado (con submenú para cambiar alguno de los atributos nivelAtaque, nivelDefensa, vidaActual)
- 5) Comparar Soldados (verifica si atributos: nombre, nivelAtaque, nivelDefensa, vidaActual y vive son iguales)
- 6) Intercambiar Soldados (intercambia 2 soldados en sus posiciones en la estructura de datos del ejército)
- 7) Ver soldado (Búsqueda por nombre)
- 8) Ver ejército
- 9) Sumar niveles (usando Method-Call Chaining), calcular las sumatorias de nivelVida, nivelAtaque, nivelDefensa, velocidad de todos los soldados de un ejército
- 1. Por ejemplo, si ejército tendría 3 soldados:
- 2. `s=s1.sumar(s2).sumar(s3);`
- 3. s es un objeto Soldado nuevo que contendría las sumatorias de los 4 atributos indicados de los 3 soldados.
- Ningún soldado cambia sus valores.
- 10) Jugar (se empezará el juego con los cambios realizados) y con las mismas opciones de la opción 1.
- 11) Volver (muestra el menú principal)
- Después de escoger alguna de las opciones 1) a 9) se podrá volver a elegir uno de los ejércitos y se mostrarán las opciones 1) a 11)

1.3. 3. Salir

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Home Single Language 64 bits 22H2.2283
- VIM 9.0.
- Visual Studio Code 64 bits 1.82.2
- OpenJDK 64-Bits 11.0.16.1
- Git 2.41.0.windows.1
- IntelliJ IDEA 2023.3 Runtime version: 17.0.9+7-b1087.7 amd64
- Cuenta en GitHub con el correo institucional.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/RyanValdivia/fp2-23b.git>
- URL para el laboratorio 12 en el Repositorio GitHub.
- <https://github.com/RyanValdivia/fp2-23b/tree/main/fase02/lab12>

4. Actividades

4.1. Actividad 1

- En primer lugar, en el primer commit, me salté a establecer y reciclar la clase Soldier de laboratorios anteriores, pero añadiendo algunos atributos y métodos para facilitarme el trabajo.

Listing 1: Obteniendo la clase Soldado

```
$ git log lab12
commit 162150fdac9e2f7d79f49848fff7967556d28945
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date: Sat Dec 9 15:24:55 2023 -0500
    Estableciendo la clase Soldier, junto a sus constructores, getters y setters
```

- Conteniendo el siguiente código de la clase Soldier

Listing 2: Clase Soldado

```
6 public class Soldier {
7     private String name;
8     private int atk;
9     private int def;
10    private int cHP;
11    private int maxHP;
12    private String flag;
13    private boolean alive;
14    private int id;
15    private int row;
16    private int column;
17
18    /**
19     * Genera un 'placeholder' de soldado, es decir, un soldado vacio
20     */
21
22    Soldier() {
23        this.name = " ";
24        this.flag = " ";
25        this.alive = false;
26    }
27
28    /**
29     * Genera un soldado con estadísticas aleatorias
30     */
```

```
31
32 Soldier(int i, int id) {
33     this.name = "Soldado" + i + "X" + id;
34     int v = (int) (Math.random() * 5) + 1;
35     int a = (int) (Math.random() * 5) + 1;
36     int d = (int) (Math.random() * 5) + 1;
37
38     this.maxHP = v;
39     this.cHP = v;
40     this.atk = a;
41     this.def = d;
42     this.id = id;
43
44     this.alive = true;
45     if (id == 1) {
46         this.flag = "#####";
47     } else {
48         this.flag = "*****";
49     }
50 }
51
52 /** Genera un soldado con estadísticas específicas */
53
54 Soldier(int a, int d, int hp, int i, int id) {
55     this.name = "Soldado" + i + "X" + id;
56
57     this.atk = a;
58     this.def = d;
59     this.cHP = hp;
60     this.maxHP = hp;
61     this.id = id;
62
63     if (id == 1) {
64         this.flag = "#####";
65     } else {
66         this.flag = "*****";
67     }
68 }
69
70 public void setPosition(int y, int x) {
71     this.row = y;
72     this.column = x;
73 }
74
75 public int getRow() {
76     return this.row;
77 }
78
79 public int getColumn() {
80     return this.column;
81 }
82
83 public void setHP(int v) {
84     this.cHP = v;
85 }
86
```

```
87     public int getcHP() {
88         return this.cHP;
89     }
90
91     public int getMaxHP() {
92         return this.maxHP;
93     }
94
95     public boolean getStatus() {
96         return this.alive;
97     }
98
99     public String getFlag() {
100         return this.flag;
101     }
102
103     public String getName() {
104         return this.name;
105     }
106
107     public int getDefense() {
108         return this.def;
109     }
110
111     public void die() {
112         this.name = " ";
113         this.flag = " ";
114         this.maxHP = 0;
115         this.def = 0;
116         this.alive = false;
117     }
118
119     public int getId() {
120         return this.id;
121     }
122
123     public void copy(Soldier s) {
124         this.name = s.name;
125         this.flag = s.flag;
126         this.def = s.def;
127         this.setPosition(s.getColumn(), s.getRow());
128         this.maxHP = s.maxHP;
129         this.cHP = s.cHP;
130         this.id = s.id;
131     }
132
133     public void curar(int n) {
134         this.cHP = this.cHP + n;
135     }
136
137     public void setName(String n) {
138         this.name = n;
139     }
140
141     public void setAtk(int a) {
142         this.atk = a;
```

```
143     }
144
145     public void setDef(int d) {
146         this.def = d;
147     }
148
149     public void add(Soldier s) {
150         this.atk += s.atk;
151         this.def += s.def;
152         this.maxHP += s.maxHP;
153     }
154
155     public int getAtk() {
156         return this.atk;
157     }
158 }
```

- Esta clase contiene 3 constructores y diferentes atributos que harán que la tarea de gestionar los datos internos del juego, sea mucho más sencilla.
- El constructor normal, hace que los atributos 'name' y 'flag' estén vacíos, para cuando llegue el momento de imprimir el tablero de juego. El segundo constructor, tendrá la tarea de crear Soldados con estadísticas aleatorias, dándole solo el id del soldado y el número del ejército al que pertenece. Por último, el último constructor será utilizado para crear Soldados con estadísticas ya definidas por el usuario.
- Ahora, yendo por el segundo commit, inicié con el trabajo para el modo de 'Partida rápida' videojuego, estableciendo métodos para inicializar cada ejército.
- Además, creé las estructuras de datos para probar estos métodos.

Listing 3: Primeros métodos

```
159     public static void initArmy(ArrayList<Soldier> ar, int nro, Soldier[][] tb) {
160         int q = (int) (Math.random() * 10) + 1;
161         for (int i = 0; i < q; i++) {
162             ar.add(new Soldier(i, nro));
163         }
164         for (int i = 0; i < ar.size(); i++) {
165             int x;
166             int y;
167             do {
168                 y = (int) (Math.random() * 10);
169                 x = (int) (Math.random() * 10);
170             } while (tb[y][x].getStatus());
171             ar.get(i).setPosition(y, x);
172             tb[y][x] = ar.get(i);
173         }
174     }
```

- Este método inicializará cada ejército (En este caso, estaré usando ArrayList para los ejércitos), toma como entrada un ArrayList (el ejército), un arreglo bidimensional de Soldados (el tablero) y un entero que determinará qué ejército es.

- El método primero inicializa cada Soldado y lo añade a la Lista, luego de ello, lo despliega en alguna coordenada del tablero (pseudoaleatoria) solo revisando que no haya ya algún soldado en dicha coordenada, si no es así, vuelve a generarla.
- Esto lo hace gracias al método de la clase Soldier 'getStatus' que nos dice si hay algún soldado vivo en esa casilla.
- Ahora, el siguiente método fue reciclado de laboratorios anteriores para mostrar el tablero una vez ya inicializado y desplegado cada ejército.

Listing 4: Show Table and Init Table

```
175 public static void initTable(Soldier[] [] tb) {
176     for (int i = 0; i < tb.length; i++) {
177         for (int j = 0; j < tb[i].length; j++) {
178             tb[i][j] = new Soldier();
179         }
180     }
181 }
182
183
184 public static void showTable(Soldier[] [] tb) {
185     String vacio = " ";
186     System.out.println(crearTecho());
187     for (int i = 0; i < tb.length; i++) {
188         System.out.println(separadorSup());
189         for (int j = 0; j < tb[i].length; j++) {
190             if (j == tb[i].length - 1) {
191                 System.out.print("| " + tb[i][j].getFlag() + " |\n");
192             } else {
193                 System.out.print("| " + tb[i][j].getFlag() + " ");
194             }
195         }
196         for (int j = 0; j < tb[i].length; j++) {
197             if (j == tb[i].length - 1) {
198                 System.out.print("| " + tb[i][j].getName() + " |\n");
199             } else {
200                 System.out.print("| " + tb[i][j].getName() + " ");
201             }
202         }
203         for (int j = 0; j < tb[i].length; j++) {
204             if (tb[i][j].getMaxHP() != 0) {
205                 if (j == tb[i].length - 1) {
206                     System.out
207                         .print("| " + tb[i][j].getcHP() + "/" + tb[i][j].getMaxHP() + "
208                             HP" + " |\n");
209                 } else {
210                     System.out.print("| " + tb[i][j].getcHP() + "/" + tb[i][j].getMaxHP() +
211                         " HP" + " ");
212                 }
213             } else {
214                 if (j == tb[i].length - 1) {
215                     System.out.print("| " + vacio + " |\n");
216                 } else {
217                     System.out.print("| " + vacio + " ");
218                 }
219             }
220         }
221     }
222 }
```

```
217     }
218   }
219   for (int j = 0; j < tb[i].length; j++) {
220     if (tb[i][j].getDefense() != 0) {
221       if (j == tb[i].length - 1) {
222         System.out.print("| " + tb[i][j].getDefense() + " DP" + " |\n");
223       } else {
224         System.out.print("| " + tb[i][j].getDefense() + " DP" + " ");
225       }
226     } else {
227       if (j == tb[i].length - 1) {
228         System.out.print("| " + vacio + " |\n");
229       } else {
230         System.out.print("| " + vacio + " ");
231       }
232     }
233   }
234   System.out.println(separadorInf());
235 }
236 System.out.println();
237 }
238
239 public static String crearTecho() {
240   String franky = "";
241   for (int i = 0; i < 131; i++) {
242     franky += "_";
243   }
244   return franky;
245 }
246
247 public static String separadorInf() {
248   String franky = "";
249   for (int i = 0; i < 131; i++) {
250     if (i % 13 == 0) {
251       System.out.print("|");
252     } else {
253       System.out.print("_");
254     }
255   }
256   return franky;
257 }
258
259 public static String separadorSup() {
260   String franky = "";
261   for (int i = 0; i < 131; i++) {
262     if (i % 13 == 0) {
263       System.out.print("|");
264     } else {
265       System.out.print(" ");
266     }
267   }
268   return franky;
269 }
```

- Este imprimirá las estadísticas de cada soldado, como el nombre, la vida, su defensa y una bandera distintiva de cada ejército para diferenciarlos. Para esto quería utilizar el constructor

vacío, para usarlo en los espacios vacíos. El cual utiliza el método 'initTable' para inicializar todo el tablero.

- Una vez este paso terminado, y moviendonos al siguiente commit, empecé a trabajar en que el juego fuera funcional, con el sistema de batallas 1 contra 1.
- Para esto, decidí manejar y crear un método 'turn' para manejar cada turno por jugador.

Listing 5: Método turn

```
270 public static void turn(Soldier[][] tb, ArrayList<Soldier> a, int[] c, int id) {
271     Scanner sc = new Scanner(System.in);
272     System.out.println("Turno del ejercito " + id + " : ");
273     System.out.println("Seleccione las coordenadas del soldado que movera (x, y): ");
274     int x, y;
275     do {
276         x = sc.nextInt();
277         y = sc.nextInt();
278         if (tb[y][x].getId() != id) {
279             System.out.println("Elige un soldado de tu propio ejercito!");
280         }
281     } while (tb[y][x].getId() != id);
282     System.out.println("Soldado a mover: ");
283     showSoldier(tb[y][x]);
284     System.out.println("Seleccione las coordenadas hacia donde se movera su soldado: ");
285     int x1, y1;
286     do {
287         x1 = sc.nextInt();
288         y1 = sc.nextInt();
289         if (Math.abs(x1 - x) > 1 || Math.abs(y1 - y) > 1) {
290             System.out.println("Solo te puedes mover una casilla!");
291         }
292         if (x1 > 9 || y1 > 9) {
293             System.out.println("Ingresa coordenadas dentro del tablero!");
294         }
295         if (tb[y1][x1].getId() == id) {
296             System.out.println("No puedes atacar a tu propio ejercito!");
297         }
298     } while (Math.abs(x1 - x) > 1 || Math.abs(y1 - y) > 1 || x1 > 9 || y1 > 9 ||
299             tb[y1][x1].getId() == id);
300     if (tb[y1][x1].getId() == id) {
301         System.out.println("No puedes atacar a tu propio ejercito!")
302     }
303     if (tb[y1][x1].getStatus()) {
304         tb[y1][x1].copy(battle(tb[y][x], tb[y1][x1]));
305         tb[y1][x1].curar(1);
306         tb[y][x].die();
307         c[0]--;
308     } else {
309         tb[y1][x1].copy(tb[y][x]);
310         tb[y][x].die();
311     }
312     showTable(tb);
313 }
```

- En este método coloqué todas las especificaciones que eran necesarias para que el juego funcione. Como que al momento de pedir las coordenadas del soldado a mover cada turno, este soldado sea de tu propio ejército (en el turno del ejército 1, no podrás mover un soldado del ejército 2)
- Además, que las coordenadas del destino del Soldado no sean más de una casilla, para que el juego sea más balanceado. Como también evitar el fuego aliado y que puedas atacar a un soldado de tu propio ejército.
- Así mismo, hice que una vez que se seleccione un soldado se muestre para el usuario.
- Ahora, lo más interesante es cuando un soldado se mueve a una casilla con otro soldado del ejército enemigo, en este caso, se da lugar a una batalla, para el cual usé otro método, el método 'battle' el cual devuelve al soldado ganador de la batalla.

Listing 6: Método battle

```
314 public static Soldier battle(Soldier s1, Soldier s2) {
315     int max = s1.getcHP() + s2.getcHP();
316     double r = Math.random();
317     if (s1.getcHP() < s2.getcHP()) {
318         if (r < s1.getcHP() / max) {
319             /* Gana s1 */
320             System.out.println("Gana " + s1.getName());
321             return s1;
322         }
323     } else {
324         /* Gana s2 */
325         System.out.println("Gana " + s2.getName());
326         return s2;
327     }
328 } else {
329     if (r < s2.getcHP() / max) {
330         /* Gana s2 */
331         System.out.println("Gana " + s2.getName());
332         return s2;
333     } else {
334         /* Gana s1 */
335         System.out.println("Gana " + s1.getName());
336         return s1;
337     }
338 }
339 }
340 }
```

- Este método indica cual de los dos soldados ha ganado y devuelve el Soldado Ganador. (La métrica es pseudoaleatoria en base a la vida de cada soldado)
- Una vez que se obtiene el soldado ganador, el proceso que se sigue es el siguiente:
- Primero, dos soldados batallan en la misma casilla, en esa casilla se establecerá el soldado ganador, usando el método 'copy' de la clase Soldier que copia todas las características de un soldado a otro, entonces la casilla pasa a copiar al Soldado Ganador.
- Luego de ello, uso el método 'die' en el soldado que se movió para que la casilla desde la que se movió quede vacía. Y listo.

- Eso ocurre si hay un soldado en la casilla de destino, si no lo hay, únicamente la casilla de destino copia al soldado y el soldado usa 'die' para dejar la casilla de inicio vacía.
- Ahora, en el método 'turn' utilicé otro método que nos servirá mucho posteriormente, el método 'showSoldier' que mostrará todos los datos de un soldado específico.

Listing 7: Show Soldier

```
341 public static void showSoldier(Soldier s) {  
342     System.out.println("Nombre: " + s.getName());  
343     System.out.println("Vida: " + s.getcHP() + " / " + s.getMaxHP() + " HP");  
344     System.out.println("Defensa: " + s.getDefense() + " DP");  
345     System.out.println("Posicion (x, y): " + s.getColumn() + " - " + s.getRow() + "\n");  
346 }
```

- Ahora que tenemos el funcionamiento de cada turno, elaboré el método principal para el juego, el método 'gameStart' donde se colocará la lógica primordial del videojuego.

Listing 8: Game Start

```
347 public static void gameStart(Soldier[] [] tb, ArrayList<Soldier> a1, ArrayList<Soldier> a2)  
348 {  
349     Scanner sc = new Scanner(System.in);  
350     int q1[] = new int[] { a1.size() };  
351     int q2[] = new int[] { a2.size() };  
352     while (q1[0] != 0 || q2[0] != 0) {  
353         System.out.println("Presiona cualquier boton para continuar: ");  
354         if (sc.next().equals("q")) {  
355             break;  
356         }  
357         turn(tb, a1, q1, 1);  
358         if (q1[0] == 0) {  
359             System.out.println("El ejercito 2 gano!");  
360             break;  
361         } else if (q2[0] == 0) {  
362             System.out.println("El ejercito 1 gano!");  
363             break;  
364         }  
365         turn(tb, a2, q2, 2);  
366         System.out.println("Presiona cualquier boton para continuar: ");  
367         if (sc.next().equals("q")) {  
368             break;  
369         }  
370         if (q1[0] == 0) {  
371             System.out.println("El ejercito 2 gano!");  
372             break;  
373         } else if (q2[0] == 0) {  
374             System.out.println("El ejercito 1 gano!");  
375             break;  
376         }  
377     }  
}
```

- En este método, tomamos dos valores que representan la cantidad de soldados vivos en cada ejército (la cual se irá disminuyendo conforme vayan muriendo soldados) y toma un bucle hasta

que alguna de estas cantidades sea 0, por cada bucle habrán dos turnos, uno para cada jugador y cada turno, se tendrá la oportunidad de cancelar el juego y volver al menú, usando la tecla 'q'.

- Una vez que ya tenemos el funcionamiento principal, falta añadir todos los métodos necesarios para el menú de opciones, tanto del Juego Rápido, como los del Juego Personalizado.
- Para el modo de partida rápida, simplemente reutilicé los métodos de laboratorios anteriores aplicándolos ahora para ArrayList.

Listing 9: Métodos reciclados

```
378 public static void sMaxHP(ArrayList<Soldier> s, int id) {
379     int max = 0;
380     for (int i = 0; i < s.size(); i++) {
381         if (s.get(i).getMaxHP() > s.get(max).getMaxHP()) {
382             max = i;
383         }
384     }
385     System.out.println("El soldado con mayor vida del ejercito " + id + " es:");
386     showSoldier(s.get(max));
387 }
388
389 public static void avgAndTotal(ArrayList<Soldier> s, int id) {
390     Soldier a = new Soldier();
391     for (int i = 0; i < s.size(); i++) {
392         a.add(s.get(i));
393     }
394     double n = a.getMaxHP() / s.size();
395     System.out.println("El total de vida del ejercito " + id + " es: " + a.getMaxHP());
396     System.out.println("La vida promedio del ejercito " + id + " es: " + n);
397 }
398
399 public static void bubbleSort(ArrayList<Soldier> a) {
400     for (int i = 0; i < a.size(); i++) {
401         for (int j = 0; j < a.size() - 1; j++) {
402             if (a.get(j).getMaxHP() < a.get(j + 1).getMaxHP()) {
403                 intercambiar(a, j, j + 1);
404             }
405         }
406     }
407 }
408
409 public static void insertionSort(ArrayList<Soldier> a) {
410     for (int i = 0; i < a.size(); i++) {
411         Soldier s = a.get(i);
412         int j = i;
413         for (j = 1; 0 < j && a.get(j - 1).getMaxHP() < s.getMaxHP(); j--) {
414             a.set(j, a.get(j - 1));
415         }
416         a.set(j, s);
417     }
418 }
419
420 public static void ranking(ArrayList<Soldier> a, int id) {
421     Scanner sc = new Scanner(System.in);
422     System.out.println("Que algoritmo de ordenamiento le gustaria usar?");
```

```
423     System.out.println("1. Bubble Sort");
424     System.out.println("2. Insertion Sort");
425     switch (sc.nextInt()) {
426         case 1:
427             bubbleSort(a);
428             showArmy(a, id);
429             break;
430         case 2:
431             insertionSort(a);
432             showArmy(a, id);
433             break;
434         default:
435             break;
436     }
437 }
438
439 public static void intercambiar(ArrayList<Soldier> a, int i, int j) {
440     Soldier temp;
441     temp = a.get(i);
442     a.set(i, a.get(j));
443     a.set(j, temp);
444 }
```

- Siendo sus utilidades, mostrar el soldado con mayor vida de cada ejército, mostrar la vida total y promedio de cada ejército, los algoritmos de ordenamiento (burbuja e inserción) y el ranking de soldados por nivel de vida, respectivamente.
- Esto sería colocado como menú utilizando una estructura do-while en el método main, para que el usuario pueda navegar libremente y usar las opciones que más le parezcan.

Listing 10: Menú de Partida Rápida

```
445  /* Juego normal */
446      int o1;
447      do {
448          System.out.println("Presiona 'q' en cualquier momento para salir");
449
450          Soldier[][] table = new Soldier[10][10];
451          initTable(table);
452
453          ArrayList<Soldier> ej1 = new ArrayList<Soldier>();
454          initArmy(ej1, 1, table);
455          ArrayList<Soldier> ej2 = new ArrayList<Soldier>();
456          initArmy(ej2, 2, table);
457
458          showTable(table);
459
460          gameStart(table, ej1, ej2);
461
462          System.out.println("Juego terminado");
463          System.out.println("Presiona 1 para volver al menu principal");
464          System.out.println("Presiona 2 para volver a jugar");
465          System.out.println("Presiona 3 para otras opciones: ");
466          o1 = sc.nextInt();
467          int o2;
```

```
468         do {
469             System.out.println("1. Soldado con mayor vida");
470             System.out.println("2. Promedio de vida y total");
471             System.out.println("3. Datos de todos los soldados");
472             System.out.println("4. Ranking de poder");
473             System.out.println("5. Salir");
474             o2 = sc.nextInt();
475             switch (o2) {
476                 case 1:
477                     System.out.println("Seleccione ejercito: ");
478                     if (sc.nextInt() == 1) {
479                         sMaxHP(ej1, 1);
480                     } else {
481                         sMaxHP(ej2, 2);
482                     }
483                     break;
484                 case 2:
485                     System.out.println("Seleccione ejercito: ");
486                     if (sc.nextInt() == 1) {
487                         avgAndTotal(ej1, 1);
488                     } else {
489                         avgAndTotal(ej2, 2);
490                     }
491                     break;
492                 case 3:
493                     System.out.println("Seleccione ejercito: ");
494                     if (sc.nextInt() == 1) {
495                         showArmy(ej1, 1);
496                     } else {
497                         showArmy(ej2, 2);
498                     }
499                     break;
500                 case 4:
501                     System.out.println("Seleccione ejercito: ");
502                     if (sc.nextInt() == 1) {
503                         ranking(ej1, 1);
504                     } else {
505                         ranking(ej2, 2);
506                     }
507                     break;
508                 default:
509                     break;
510             }
511             } while (o2 != 5);
512
513     } while (o1 == 2);
```

- Una vez terminada esta parte, seguía lo más tardado, el modo personalizado. Donde también debía hacer un menú pero con muchas más opciones. Al final de todo, debía añadir diferentes métodos para su funcionamiento.

Listing 11: Método Create Soldier

```
514     public static void createSoldier(ArrayList<Soldier> a, int id, Soldier[][] tb) {
515         Scanner sc = new Scanner(System.in);
```

```
516     int atk, hp, def, x, y;
517     System.out.println("Ingrese los datos de su soldado");
518     System.out.println("Ataque: ");
519     atk = sc.nextInt();
520     System.out.println("Vida maxima: ");
521     hp = sc.nextInt();
522     System.out.println("Defensa: ");
523     def = sc.nextInt();
524     System.out.println("Posicion (x, y): ");
525     do {
526         x = sc.nextInt();
527         y = sc.nextInt();
528         if (tb[y][x].getStatus()) {
529             System.out.println("Ya hay un soldado en ese lugar!");
530         }
531     } while (tb[y][x].getStatus());
532
533     Soldier s = new Soldier(atk, def, hp, a.size() - 1, id);
534     s.setPosition(y, x);
535     tb[y][x].copy(s);
536 }
```

- Este método creará un Soldado con las estadísticas que el usuario desee, en las coordenadas deseadas, revisando también que no haya dos soldados en la misma casilla y que tampoco haya más de 10 soldados en un solo ejército.

Listing 12: Más métodos

```
537 public static void deleteSoldier(Soldier[] [] tb) {
538     Scanner sc = new Scanner(System.in);
539     System.out.println("Seleccione las coordenadas del soldado a eliminar (x, y)");
540     int x = sc.nextInt();
541     int y = sc.nextInt();
542     tb[y][x].die();
543 }
544
545 public static int searchSoldier(ArrayList<Soldier> a) {
546     Scanner sc = new Scanner(System.in);
547     System.out.println("Ingrese el nombre del soldado: ");
548     String n = sc.next();
549     for (int i = 0; i < a.size(); i++) {
550         if (n.equals(a.get(i).getName())) {
551             return i;
552         }
553     }
554     return -1;
555 }
556
557 public static void cloneSoldier(ArrayList<Soldier> a, int id, Soldier[] [] tb) {
558     Scanner sc = new Scanner(System.in);
559     int i = searchSoldier(a);
560     int x, y;
561     System.out.println("Seleccione las coordenadas a las que se desplegara la copia (x, y): ");
562     do {
```

```
563         x = sc.nextInt();
564         y = sc.nextInt();
565         if (tb[y][x].getStatus()) {
566             System.out.println("Ya hay un soldado en ese lugar!");
567         }
568     } while (tb[y][x].getStatus());
569     Soldier s = new Soldier();
570     s.copy(a.get(i));
571     s.setPosition(y, x);
572     s.setName("Soldado" + (a.size() - 1) + "X" + id);
573     a.add(s);
574 }
```

- Siguen tres métodos, el primero eliminará un soldado, esto es algo sencillo usando el método 'die' de la clase Soldier.
- El segundo método realiza una búsqueda lineal en todo un ejército para encontrar un soldado en específico, devolviendo su índice en el ArrayList, realiza una búsqueda por nombre del soldado.
- Por último, la opción para clonar el soldado a elección por el usuario y añadirlo al final de la lista.
- Lo siguiente era avanzar con la opción de modificarse intercambiar soldados en el tablero, de igual forma hice varios métodos para solucionar esto.

Listing 13: Métodos especiales

```
575 public static void modifySoldier(ArrayList<Soldier> a, int id, Soldier[][] tb) {
576     Scanner sc = new Scanner(System.in);
577     System.out.println("Ingresa el nombre del soldado a modificar: ");
578     int i = searchSoldier(a);
579     showSoldier(a.get(i));
580     System.out.println("Que estadística desea modificar?");
581     System.out.println("1. Ataque");
582     System.out.println("2. Defensa");
583     System.out.println("3. Vida");
584     switch (sc.nextInt()) {
585         case 1:
586             System.out.println("Ingresa la nueva cantidad: ");
587             a.get(i).setAtk(sc.nextInt());
588             break;
589         case 2:
590             System.out.println("Ingresa la nueva cantidad: ");
591             a.get(i).setDef(sc.nextInt());
592             break;
593         case 3:
594             System.out.println("Ingresa la nueva cantidad: ");
595             a.get(i).setHP(sc.nextInt());
596             break;
597         default:
598             break;
599     }
600 }
601
602 public static void interchange(Soldier[][] tb) {
```



```
603 Scanner sc = new Scanner(System.in);
604 Soldier s = new Soldier();
605 System.out.println("Ingrese las coordenadas del primer soldado (x, y)");
606 int x, y, a, b;
607 do {
608     x = sc.nextInt();
609     y = sc.nextInt();
610     if (!tb[y][x].getStatus()) {
611         System.out.println("No hay ningun soldado alli!");
612     }
613 } while (!tb[y][x].getStatus());
614 System.out.println("Ingrese las coordenadas del segundo soldado");
615 do {
616     a = sc.nextInt();
617     b = sc.nextInt();
618     if (!tb[b][a].getStatus()) {
619         System.out.println("No hay ningun soldado alli!");
620     }
621 } while (!tb[b][a].getStatus());
622 s.copy(tb[y][x]);
623 tb[y][x].copy(tb[b][a]);
624 tb[b][a].copy(s);
625 }
```

- En el primer método, pido al usuario las coordenadas del soldado a modificar y las cantidades nuevas de cada estadística, con su respectivo menú.
- En el segundo, pido las coordenadas de los dos soldados y luego creo una variable temporal para hacer el intercambio.
- El siguiente método por hacer era el de comparación.

Listing 14: Compare Soldier

```
626 public static void compareSoldiers(Soldier[][] tb, int id) {
627     Scanner sc = new Scanner(System.in);
628     System.out.println("Ingrese las coordenadas del 1er soldado a comparar");
629     int x, y, a, b;
630     do {
631         x = sc.nextInt();
632         y = sc.nextInt();
633         if (!tb[y][x].getStatus()) {
634             System.out.println("No hay ningun soldado alli!");
635         }
636     } while (!tb[y][x].getStatus());
637     Soldier s1 = tb[y][x];
638     System.out.println("Ingrese las coordenadas del 2do soldado a comparar");
639     do {
640         a = sc.nextInt();
641         b = sc.nextInt();
642         if (!tb[b][a].getStatus()) {
643             System.out.println("No hay ningun soldado alli!");
644         }
645     } while (!tb[b][a].getStatus());
646     Soldier s2 = tb[b][a];
647     switch (id) {
```

```
648         case 1:
649             if (s1.getName().compareTo(s2.getName()) != 0) {
650                 System.out.println("No son iguales");
651             }
652             break;
653         case 2:
654             if (s1.getAtk() != s2.getAtk()) {
655                 System.out.println("No son iguales");
656             }
657             break;
658         case 3:
659             if (s2.getDefense() != s2.getDefense()) {
660                 System.out.println("No son iguales");
661             }
662             break;
663         case 4:
664             if (s1.getMaxHP() != s2.getMaxHP()) {
665                 System.out.println("No son iguales");
666             }
667             break;
668         case 5:
669             if (s1.getStatus() != s2.getStatus()) {
670                 System.out.println("No son iguales");
671             }
672         default:
673             break;
674     }
675 }
```

- En este método se piden los dos soldados a comparar y se usa un menú para ver que estadística comparar.
- Luego, quedaban los métodos más sencillos, como ver un soldado (buscandolo en base a nombre), mostrar todo el ejército, sumar niveles, jugar y volver al menú principal. Algunos los pude reciclar de anteriores laboratorios.

Listing 15: Otros

```
676 public static void seeSoldier(ArrayList<Soldier> a) {
677     int i = searchSoldier(a);
678     showSoldier(a.get(i));
679 }
680 public static void addLevels(ArrayList<Soldier> a) {
681     Soldier s = new Soldier();
682     for (int i = 0; i < a.size(); i++) {
683         s.add(a.get(i));
684     }
685     System.out.println("El nivel total de vida es: " + s.getMaxHP());
686     System.out.println("El nivel total de defensa es: " + s.getDefense());
687     System.out.println("El nivel total de ataque es: " + s.getAtk());
688 }
689 public static void showSoldier(Soldier s) {
690     System.out.println("Nombre: " + s.getName());
691     System.out.println("Vida: " + s.getcHP() + " / " + s.getMaxHP() + " HP");
692     System.out.println("Defensa: " + s.getDefense() + " DP");
693 }
```

```
693     System.out.println("Posicion (x, y): " + s.getColumn() + " - " + s.getRow() + "\n");
694 }
695
696 public static void showArmy(ArrayList<Soldier> s, int id) {
697     System.out.println("Ejercito " + id + "\n");
698     for (int i = 0; i < s.size(); i++) {
699         showSoldier(s.get(i));
700         System.out.println();
701     }
702 }
```

- Con todo esto ya hecho, solo queda ensamblar todo, armar los menús para el usuario y las opciones, todo en el método main. Lo cual no fue muy difícil, usé varias estructuras do while para poder hacer recursivo el menú. Así quedaría todo armado para que el juego funcione:

Listing 16: Método main Final

```
703 public static void main(String[] args) {
704     Scanner sc = new Scanner(System.in);
705     int option;
706     do {
707         System.out.println("Bienvenido! Que quieres jugar?");
708         System.out.println("1. Juego rapido");
709         System.out.println("2. Juego personalizado");
710         System.out.println("3. Salir");
711         option = sc.nextInt();
712         if (option == 3) {
713             break;
714         }
715         switch (option) {
716             case 1:
717                 /* Juego normal */
718                 int o1;
719                 do {
720                     System.out.println("Presiona 'q' en cualquier momento para salir");
721
722                     Soldier[] [] table = new Soldier[10][10];
723                     initTable(table);
724
725                     ArrayList<Soldier> ej1 = new ArrayList<Soldier>();
726                     initArmy(ej1, 1, table);
727                     ArrayList<Soldier> ej2 = new ArrayList<Soldier>();
728                     initArmy(ej2, 2, table);
729
730                     showTable(table);
731
732                     gameStart(table, ej1, ej2);
733
734                     System.out.println("Juego terminado");
735                     System.out.println("Presiona 1 para volver al menu principal");
736                     System.out.println("Presiona 2 para volver a jugar");
737                     System.out.println("Presiona 3 para otras opciones: ");
738                     o1 = sc.nextInt();
739                     int o2;
740                     do {
```

```
741         System.out.println("1. Soldado con mayor vida");
742         System.out.println("2. Promedio de vida y total");
743         System.out.println("3. Datos de todos los soldados");
744         System.out.println("4. Ranking de poder");
745         System.out.println("5. Salir");
746         o2 = sc.nextInt();
747         switch (o2) {
748             case 1:
749                 System.out.println("Seleccione ejercito: ");
750                 if (sc.nextInt() == 1) {
751                     sMaxHP(ej1, 1);
752                 } else {
753                     sMaxHP(ej2, 2);
754                 }
755                 break;
756             case 2:
757                 System.out.println("Seleccione ejercito: ");
758                 if (sc.nextInt() == 1) {
759                     avgAndTotal(ej1, 1);
760                 } else {
761                     avgAndTotal(ej2, 2);
762                 }
763                 break;
764             case 3:
765                 System.out.println("Seleccione ejercito: ");
766                 if (sc.nextInt() == 1) {
767                     showArmy(ej1, 1);
768                 } else {
769                     showArmy(ej2, 2);
770                 }
771                 break;
772             case 4:
773                 System.out.println("Seleccione ejercito: ");
774                 if (sc.nextInt() == 1) {
775                     ranking(ej1, 1);
776                 } else {
777                     ranking(ej2, 2);
778                 }
779                 break;
780             default:
781                 break;
782         }
783     } while (o2 != 5);
784
785     } while (o1 == 2);
786     break;
787 case 2:
788     int o2;
789     /* Juego Personalizado */
790     System.out.println("Presiona 'q' en cualquier momento para salir");
791     Soldier[] [] table = new Soldier[10][10];
792     initTable(table);
793
794     ArrayList<Soldier> ej1 = new ArrayList<Soldier>();
795     initArmy(ej1, 1, table);
796     ArrayList<Soldier> ej2 = new ArrayList<Soldier>();
```

```
797         initArmy(ej2, 2, table);
798
799         playArmy(ej1, table);
800         playArmy(ej2, table);
801
802         showTable(table);
803
804         do {
805             System.out.println("Presiona 'q' en cualquier momento para salir");
806
807             showTable(table);
808
809             System.out.println("Opciones: ");
810             System.out.println("1. Crear Soldado: ");
811             System.out.println("2. Borrar Soldado: ");
812             System.out.println("3. Clonar Soldado: ");
813             System.out.println("4. Modificar Soldado: ");
814             System.out.println("5. Comparar Soldados: ");
815             System.out.println("6. Intercambiar Soldados: ");
816             System.out.println("7. Buscar Soldado: ");
817             System.out.println("8. Ver ejercito: ");
818             System.out.println("9. Sumar niveles: ");
819             System.out.println("10. Jugar: ");
820             System.out.println("11. Salir");
821
822             o2 = sc.nextInt();
823             switch (o2) {
824                 case 1:
825                     System.out.println("Seleccione ejercito: ");
826                     if (sc.nextInt() == 1) {
827                         if (ej1.size() > 10) {
828                             System.out.println("El ejercito no puede tener mas de 10
829                                 soldados");
830                         } else {
831                             createSoldier(ej1, 1, table);
832                             playArmy(ej1, table);
833                             showTable(table);
834                         }
835                     } else {
836                         if (ej2.size() > 10) {
837                             System.out.println("El ejercito no puede tener mas de 10
838                                 soldados");
839                         } else {
840                             createSoldier(ej2, 2, table);
841                             playArmy(ej2, table);
842                             showTable(table);
843                         }
844                     }
845                     break;
846                 case 2:
847                     System.out.println("Seleccione ejercito: ");
848                     if (sc.nextInt() == 1) {
849                         if (ej1.size() == 1) {
850                             System.out.println("El ejercito no puede estar vacio!");
851                         } else {
852                             deleteSoldier(table);
853                         }
854                     }
855                     break;
856                 case 3:
857                     System.out.println("Seleccione ejercito: ");
858                     if (sc.nextInt() == 1) {
859                         if (ej1.size() > 0) {
860                             deleteSoldier(ej1, sc.nextInt(), table);
861                         } else {
862                             System.out.println("El ejercito no tiene soldados");
863                         }
864                     } else {
865                         if (ej2.size() > 0) {
866                             deleteSoldier(ej2, sc.nextInt(), table);
867                         } else {
868                             System.out.println("El ejercito no tiene soldados");
869                         }
870                     }
871                     break;
872                 case 4:
873                     System.out.println("Seleccione ejercito: ");
874                     if (sc.nextInt() == 1) {
875                         System.out.println("Seleccione el indice del soldado a modificar: ");
876                         int i = sc.nextInt();
877                         if (i < 0 || i > ej1.size() - 1) {
878                             System.out.println("Indice fuera de rango");
879                         } else {
880                             System.out.println("Ingrese el nuevo valor: ");
881                             int v = sc.nextInt();
882                             ej1.get(i).set(v);
883                             showTable(table);
884                         }
885                     } else {
886                         System.out.println("Seleccione el indice del soldado a modificar: ");
887                         int i = sc.nextInt();
888                         if (i < 0 || i > ej2.size() - 1) {
889                             System.out.println("Indice fuera de rango");
890                         } else {
891                             System.out.println("Ingrese el nuevo valor: ");
892                             int v = sc.nextInt();
893                             ej2.get(i).set(v);
894                             showTable(table);
895                         }
896                     }
897                     break;
898                 case 5:
899                     System.out.println("Seleccione ejercito: ");
900                     if (sc.nextInt() == 1) {
901                         System.out.println("Seleccione el indice del soldado a comparar: ");
902                         int i = sc.nextInt();
903                         if (i < 0 || i > ej1.size() - 1) {
904                             System.out.println("Indice fuera de rango");
905                         } else {
906                             System.out.println("Ingrese el indice del otro soldado: ");
907                             int j = sc.nextInt();
908                             if (j < 0 || j > ej2.size() - 1) {
909                                 System.out.println("Indice fuera de rango");
910                             } else {
911                                 System.out.println("Comparando: ");
912                                 System.out.println("Soldado 1: " + ej1.get(i));
913                                 System.out.println("Soldado 2: " + ej2.get(j));
914                                 System.out.println("Resultado: " + compare(ej1.get(i), ej2.get(j)));
915                             }
916                         }
917                     } else {
918                         System.out.println("Seleccione el indice del soldado a comparar: ");
919                         int i = sc.nextInt();
920                         if (i < 0 || i > ej2.size() - 1) {
921                             System.out.println("Indice fuera de rango");
922                         } else {
923                             System.out.println("Ingrese el indice del otro soldado: ");
924                             int j = sc.nextInt();
925                             if (j < 0 || j > ej1.size() - 1) {
926                                 System.out.println("Indice fuera de rango");
927                             } else {
928                                 System.out.println("Comparando: ");
929                                 System.out.println("Soldado 1: " + ej2.get(i));
930                                 System.out.println("Soldado 2: " + ej1.get(j));
931                                 System.out.println("Resultado: " + compare(ej2.get(i), ej1.get(j)));
932                             }
933                         }
934                     }
935                     break;
936                 case 6:
937                     System.out.println("Seleccione ejercito: ");
938                     if (sc.nextInt() == 1) {
939                         System.out.println("Seleccione el indice del soldado a intercambiar: ");
940                         int i = sc.nextInt();
941                         if (i < 0 || i > ej1.size() - 1) {
942                             System.out.println("Indice fuera de rango");
943                         } else {
944                             System.out.println("Ingrese el indice del otro soldado: ");
945                             int j = sc.nextInt();
946                             if (j < 0 || j > ej2.size() - 1) {
947                                 System.out.println("Indice fuera de rango");
948                             } else {
949                                 System.out.println("Intercambiando: ");
950                                 System.out.println("Soldado 1: " + ej1.get(i));
951                                 System.out.println("Soldado 2: " + ej2.get(j));
952                                 System.out.println("Resultado: " + interchange(ej1.get(i), ej2.get(j)));
953                             }
954                         }
955                     } else {
956                         System.out.println("Seleccione el indice del soldado a intercambiar: ");
957                         int i = sc.nextInt();
958                         if (i < 0 || i > ej2.size() - 1) {
959                             System.out.println("Indice fuera de rango");
960                         } else {
961                             System.out.println("Ingrese el indice del otro soldado: ");
962                             int j = sc.nextInt();
963                             if (j < 0 || j > ej1.size() - 1) {
964                                 System.out.println("Indice fuera de rango");
965                             } else {
966                                 System.out.println("Intercambiando: ");
967                                 System.out.println("Soldado 1: " + ej2.get(i));
968                                 System.out.println("Soldado 2: " + ej1.get(j));
969                                 System.out.println("Resultado: " + interchange(ej2.get(i), ej1.get(j)));
970                             }
971                         }
972                     }
973                     break;
974                 case 7:
975                     System.out.println("Seleccione ejercito: ");
976                     if (sc.nextInt() == 1) {
977                         System.out.println("Ingrese el nombre del soldado a buscar: ");
978                         String n = sc.next();
979                         System.out.println("Resultado: " + search(ej1, n));
980                     } else {
981                         System.out.println("Ingrese el nombre del soldado a buscar: ");
982                         String n = sc.next();
983                         System.out.println("Resultado: " + search(ej2, n));
984                     }
985                     break;
986                 case 8:
987                     System.out.println("Seleccione ejercito: ");
988                     if (sc.nextInt() == 1) {
989                         System.out.println("Ingrese el nivel a sumar: ");
990                         int n = sc.nextInt();
991                         System.out.println("Resultado: " + sum(ej1, n));
992                     } else {
993                         System.out.println("Seleccione ejercito: ");
994                         if (sc.nextInt() == 1) {
995                             System.out.println("Ingrese el nivel a sumar: ");
996                             int n = sc.nextInt();
997                             System.out.println("Resultado: " + sum(ej2, n));
998                         }
999                     }
1000                     break;
1001                 case 9:
1002                     System.out.println("Seleccione ejercito: ");
1003                     if (sc.nextInt() == 1) {
1004                         System.out.println("Ingrese el nivel a sumar: ");
1005                         int n = sc.nextInt();
1006                         System.out.println("Resultado: " + sum(ej1, n));
1007                     } else {
1008                         System.out.println("Seleccione ejercito: ");
1009                         if (sc.nextInt() == 1) {
1010                             System.out.println("Ingrese el nivel a sumar: ");
1011                             int n = sc.nextInt();
1012                             System.out.println("Resultado: " + sum(ej2, n));
1013                         }
1014                     }
1015                     break;
1016                 case 10:
1017                     System.out.println("Seleccione ejercito: ");
1018                     if (sc.nextInt() == 1) {
1019                         System.out.println("Ingrese el nivel a sumar: ");
1020                         int n = sc.nextInt();
1021                         System.out.println("Resultado: " + sum(ej1, n));
1022                     } else {
1023                         System.out.println("Seleccione ejercito: ");
1024                         if (sc.nextInt() == 1) {
1025                             System.out.println("Ingrese el nivel a sumar: ");
1026                             int n = sc.nextInt();
1027                             System.out.println("Resultado: " + sum(ej2, n));
1028                         }
1029                     }
1030                     break;
1031                 case 11:
1032                     System.out.println("Seleccione ejercito: ");
1033                     if (sc.nextInt() == 1) {
1034                         System.out.println("Ingrese el nivel a sumar: ");
1035                         int n = sc.nextInt();
1036                         System.out.println("Resultado: " + sum(ej1, n));
1037                     } else {
1038                         System.out.println("Seleccione ejercito: ");
1039                         if (sc.nextInt() == 1) {
1040                             System.out.println("Ingrese el nivel a sumar: ");
1041                             int n = sc.nextInt();
1042                             System.out.println("Resultado: " + sum(ej2, n));
1043                         }
1044                     }
1045                     break;
1046                 default:
1047                     System.out.println("Opcion no valida");
1048             }
1049         } while (sc.next().equals("q"));
1050     }
1051 }
```

```
851         showTable(table);
852     }
853 } else {
854     if (ej2.size() == 1) {
855         System.out.println("El ejercito no puede estar vacio!");
856     } else {
857         deleteSoldier(table);
858         showTable(table);
859     }
860 }
861 break;
862 case 3:
863     System.out.println("Seleccione ejercito: ");
864     if (sc.nextInt() == 1) {
865         if (ej1.size() > 10) {
866             System.out.println("El ejercito no puede tener mas de 10
867                 soldados");
868         } else {
869             cloneSoldier(ej1, 1, table);
870             playArmy(ej1, table);
871         }
872     } else {
873         if (ej2.size() > 10) {
874             System.out.println("El ejercito no puede tener mas de 10
875                 soldados");
876         } else {
877             cloneSoldier(ej2, 2, table);
878             playArmy(ej2, table);
879         }
880     }
881     break;
882 case 4:
883     System.out.println("Seleccione ejercito: ");
884     if (sc.nextInt() == 1) {
885         modifySoldier(ej1, 1, table);
886     } else {
887         modifySoldier(ej2, 2, table);
888     }
889     break;
890 case 5:
891     System.out.println("Opciones de comparacion: ");
892     System.out.println("1. Nombre");
893     System.out.println("2. Ataque");
894     System.out.println("3. Defensa");
895     System.out.println("4. Vida");
896     System.out.println("5. Status");
897     compareSoldiers(table, sc.nextInt());
898 case 6:
899     interchange(table);
900     break;
901 case 7:
902     System.out.println("Seleccione ejercito: ");
903     if (sc.nextInt() == 1) {
904         seeSoldier(ej1);
905     } else {
906         seeSoldier(ej2);
907     }
```

```
905         }
906         break;
907     case 8:
908         System.out.println("Seleccione ejercito: ");
909         if (sc.nextInt() == 1) {
910             showArmy(ej1, 1);
911         } else {
912             showArmy(ej2, 2);
913         }
914         break;
915     case 9:
916         System.out.println("Seleccione ejercito: ");
917         if (sc.nextInt() == 1) {
918             System.out.println("Ejercito 1:");
919             addLevels(ej1);
920         } else {
921             System.out.println("Ejercito 2:");
922             addLevels(ej2);
923         }
924         break;
925     case 10:
926         gameStart(table, ej1, ej2);
927     default:
928         break;
929     }
930     } while (o2 != 11);
931     break;
932 default:
933     break;
934 }
935 } while (option != 3);
936 sc.close();
937 }
```

- Ahora, presento capturas de la ejecución del código, como el tablero y los menús.
- Asimismo, con el diagrama uml de las dos clases.

						##### Soldado3X1 2/2 HP 4 DP			
##### Soldado1X2 3/3 HP 4 DP			##### Soldado2X1 2/2 HP 5 DP			##### Soldado0X1 1/1 HP 3 DP			
##### Soldado0X2 2/2 HP 3 DP							##### Soldado3X2 3/3 HP 1 DP		
		##### Soldado0X1 5/5 HP 1 DP							
##### Soldado0X2 5/5 HP 3 DP							##### Soldado5X2 4/4 HP 5 DP		
						##### Soldado1X1 2/2 HP 3 DP			
##### Soldado0X2 1/1 HP 4 DP									
	##### Soldado6X1 4/4 HP 1 DP						##### Soldado7X1 1/1 HP 4 DP		
		##### Soldado0X1 2/2 HP 1 DP				##### Soldado5X1 4/4 HP 5 DP			

```

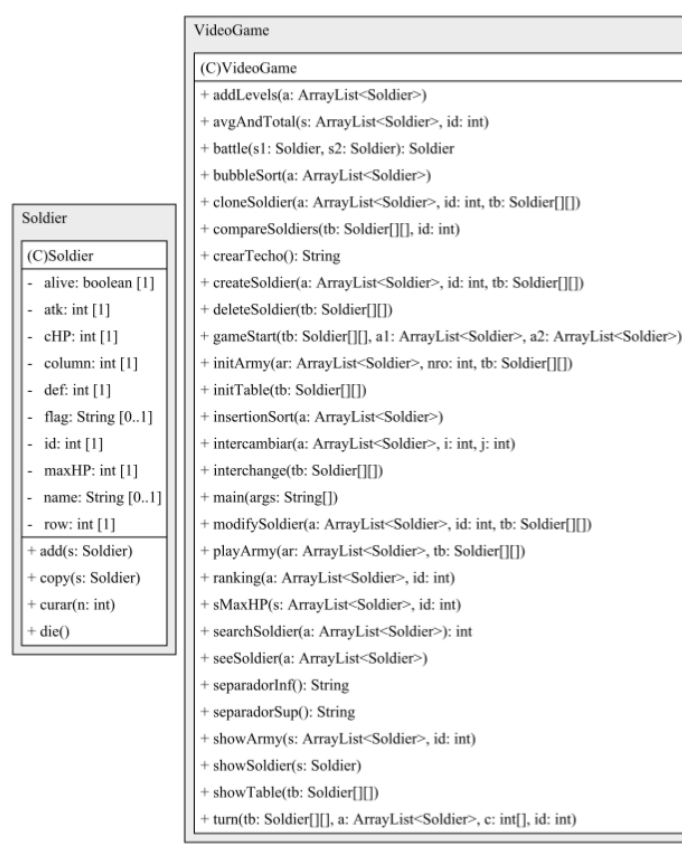
Bienvenido! Que quieres jugar?
1. Juego rapido
2. Juego personalizado
3. Salir
1
Presiona 'q' en cualquier momento para salir
    
```

```

Presiona cualquier boton para continuar:
q
Juego terminado
Presiona 1 para volver al menu principal
Presiona 2 para volver a jugar
Presiona 3 para otras opciones:
3
1. Soldado con mayor vida
2. Promedio de vida y total
3. Datos de todos los soldados
4. Ranking de poder
5. Salir
    
```


Opciones:

1. Crear Soldado:
2. Borrar Soldado:
3. Clonar Soldado:
4. Modificar Soldado:
5. Comparar Soldados:
6. Intercambiar Soldados:
7. Buscar Soldado:
8. Ver ejercito:
9. Sumar niveles:
10. Jugar:
11. Salir



5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

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	2	
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	4	
Total		20		18	

6. Referencias

- Fundamentos de la programación 2 - Tópicos de la programación Orientada a Objetos (Marco Aedo)