Universidad Nacional San Agustín de Arequipa

Facultad de Ingeniería de Producción y Servicios

Escuela Profesional de Ingeniería de Sistemas

Asignatura:

Fundamentos de la Programación 2

Tema:

Laboratorio 9

Docente:

Arroyo Revilla Christian

Estudiante:

Cuno Cahuari Armando Steven

2023

Informe de Laboratorio 9

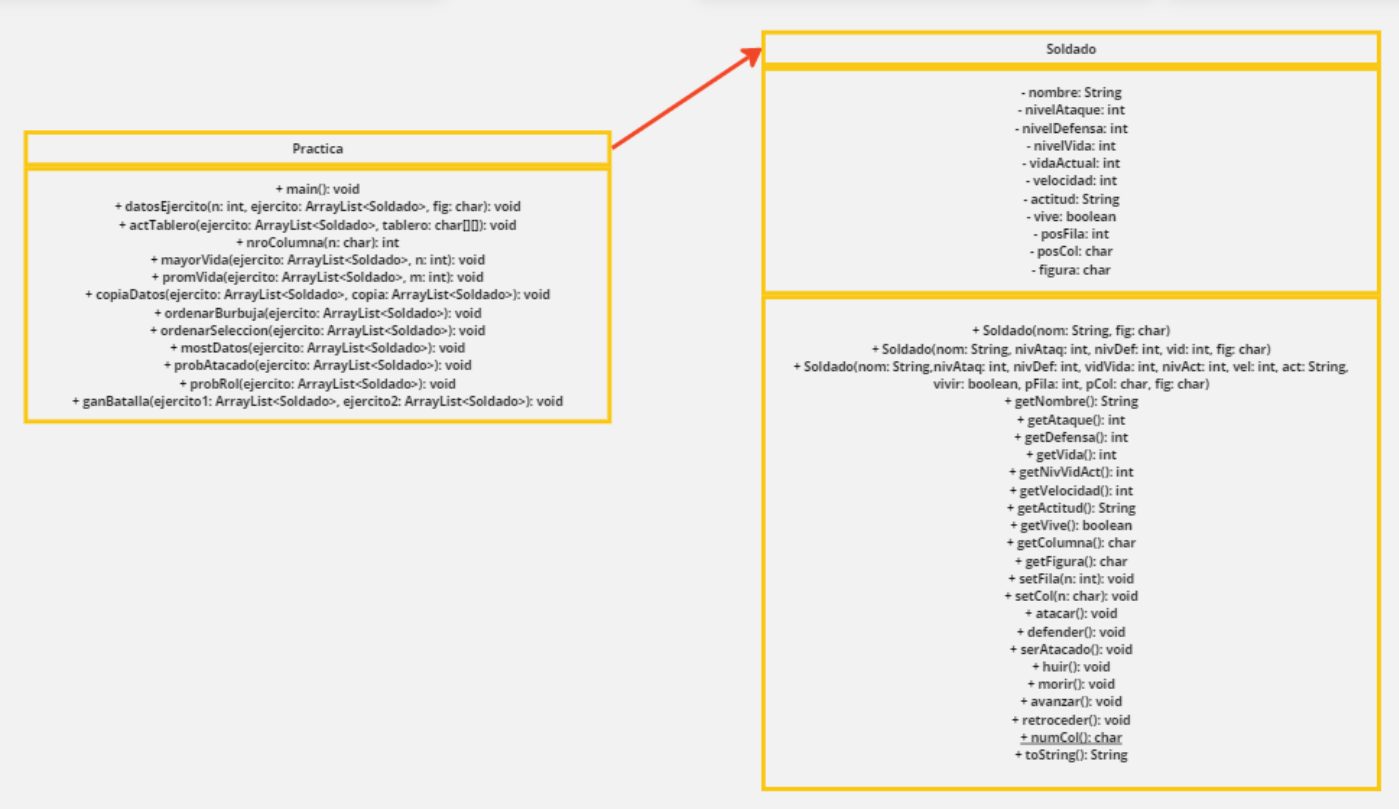
Tema: Definición Clases de Usuario

Actividad: Laboratorio 9

URL Repositorio GitHub: `//github.com/SteveArms/LaboratorioFPII.git`
Tiempo: 2 horas

Actividad:

Diagrama de Clases UML:



Clase Soldado:

Como en laboratorios pasados ya tuvimos atributos privados de encapsulamiento llamados nombre, fila, columna, vida, figura para este laboratorio hemos agregado unos cuantos más para ello asimismo creamos 3 constructores sobrecargados para la instanciación al ser usados en el main.

También métodos de instancia que son los set(nos ayudara para saber si ocupamos la posición de otro y solo modificar, ubicarlo en otro lugar) y get(Para la copia de los ArrayList Original hacia una copia). Los valores aleatorios que se vayan asignar a cada objeto Soldado se hará en el momento que hagamos instancia gracias a 2 constructores.

```java
public class Soldado {
    private String nombre;
    private int nivelAtaque;
    private int nivelDefensa;
    private int nivelVida;
    private int vidaActual;
    private int velocidad;
    private String actitud;
    private boolean vive;
    private int posFila;
    private char posCol;
    private char figura;
```

```java
    public Soldado(String nom, char fig){
        nombre = nom;
        nivelAtaque = (int)(Math.random() * 5 + 1);
        nivelDefensa = (int)(Math.random() * 5 + 1);
        nivelVida = (int)(Math.random() * 5 + 1);
        vidaActual = nivelVida;
        velocidad = 0;
        actitud = "Defensiva";
        vive = true;
        posFila = (int)(Math.random() * 10 + 1);
        posCol = numCol();
        figura = fig;
    }
    public Soldado(String nom, int nivAtaq, int nivDef, int vid, char fig){
        nombre = nom;
        nivelAtaque = nivAtaq;
        nivelDefensa = nivDef;
        nivelVida = vid;
        vidaActual = vid;
        velocidad = 0;
        actitud = "Defensiva";
        vive = true;
        posFila = (int)(Math.random() * 10 + 1);
        posCol = numCol();
        figura = fig;
    }
```

```java
    public Soldado(String nom, int nivAtaq, int nivDef, int nivVida, int nivAct, int vel, String act, boolean vivir, int pF
        nombre = nom;
        nivelAtaque = nivAtaq;
        nivelDefensa = nivDef;
        nivelVida = nivVida;
        vidaActual = nivAct;
        velocidad = vel;
        actitud = act;
        vive = vivir;
        posFila = pFila;
        posCol = pCol;
        figura = fig;
    }
```

Metodos de Instancia Get – Estos métodos, como bien se redacto, nos ayudara para entregar los valores de cada objeto Soldado. Asimismo nos ayudara para hacer una copia del Original gracias a que creamos una copia del mismo Objeto y no solo copiamos su referencia.

```java
public String getNombre(){
    return nombre;
}
public int getAtaque(){
    return nivelAtaque;
}
public int getDefensa(){
    return nivelDefensa;
}
public int getNivVidAct(){
    return vidaActual;
}
public int getVida(){
    return nivelVida;
}
public int getVelocidad(){
    return velocidad;
}
public String getActitud(){
    return actitud;
}
```

```java
public boolean getVive(){
    return vive;
}
public int getFila(){
    return posFila;
}
public char getColumna(){
    return posCol;
}
public char getFigura(){
    return figura;
}
```

Métodos set – Instanciación

Como bien se redactó esto nos ayudara para ubicar en una nueva posición si en caso, la posición donde se colocó ya está ocupada por otra figura solo configurando tanto la posición de la fila y la columna.

```java
public void setFila(int n){
    posFila = n;
}
public void setCol(char n){
    posCol = n;
}
```

Método atacar, defender, huir, morir, serAtacado, avanzar, defender

Tengamos en cuenta que la actitud del Soldado variaría si en caso ataque, defienda, o huya. Para ello tenemos la actitud Ofensiva cuando ataca; Defensiva, defiende y Fuga, huye.

Método Atacar: Como bien decía el soldado solo ataca cuando avanza y este al avanzar aumenta su velocidad en 1.

```java
public void atacar(){
    if(!getActitud().equals("Ofensiva"))
        actitud = "Ofensiva";
    avanzar();
}
```

```java
public void avanzar(){
    velocidad++;
}
```

Metodo Defender: Al defender solo el soldado se para con lo que su velocidad seria 0

```java
public void defender(){
    if(!getActitud().equals("Defensiva"))
        actitud = "Defensiva";
    velocidad = 0;
}
```

Método Huye: Al huir la velocidad aumenta en 2.

```
107     public void huir(){
108         actitud = "Fuga";
109         velocidad += 2;
110     }
```

Metodo Retroceder: Si la velocidad es mayor que 0, entonces primero su velocidad se colocaría en 0 y se pondría a la defensiva, en cambio si la velocidad es 0 este cambiara a -1.

```
118     public void retroceder(){
119         if(this.velocidad > 0){
120             velocidad = 0;
121             actitud = "Defensiva";
122         } else if(this.velocidad == 0){
123             velocidad = -1;
124         }
125     }
```

Metodo serAtacado: Al ser atacado recibirá 1 de daño asimismo este tiene la condición de que puede morir.

```
101     public void serAtacado(){
102         vidaActual--;
103         if(this.vidaActual==0){
104             morir();
105         }
106     }
```

```
111     public void morir(){
112         vive = false;
113         velocidad = 0;
114     }
```

Los métodos complementarios que se tienen en la Clase Soldado:

Metodo que retorna la posición de columna – char:

```
126     public static char numCol(){
127         String a = "abcdefghij";
128         int n = (int)(Math.random() * a.length());
129         char car = a.charAt(n);
130         return car;
131     }
```

Metodo toString:

```
132 @Override
133 public String toString(){
134     return "Nombre: " + nombre + " Vida:" + vidaActual + " Fila:" + posFila + " Columna:" + posCol + " Actitud:" + actitud;
135 }
```

Clase Main – Practica:

Empezamos creando ArrayList de objetos Soldado asimismo a este usaremos un constructor sobrecargado que lleva solo 2 parámetros. Con ayuda de otro método llamado datosEjercito que este se encarga de crear la cantidad de soldados asimismo el nombre que llevaran.

```
15          ArrayList<Soldado> ejercito_1 = new ArrayList<Soldado>();
16          datosEjercito(1, ejercito_1, '%');
17          ArrayList<Soldado> ejercito_2 = new ArrayList<Soldado>();
18          datosEjercito(2, ejercito_2, '*');
```

```
108     public static void datosEjercito(int n, ArrayList<Soldado> ejercito, char fig){
109         int nroSoldados = (int)(Math.random() * 10 + 1);
110         System.out.println("El ejercito " + n + " tiene un total de " + nroSoldados + " soldados");
111         for(int i = 0; i < nroSoldados; i++){
112             String nombre = "Soldado " + ( i + 1) + "X" + n;
113             ejercito.add(new Soldado(nombre, fig));
114         }
115 
116     }
```

Realizado esto deberemos actualizar las posiciones de estos Soldado representándolos en el tablero bidimensional de char, con ayuda del método de instancia getFigura para reemplazar el espacio indicado, por otro lado, si en caso revisa que otro Soldado quiere sobre ponerse sobre otro Soldado este deberá acudir a una nueva posición mediante el método aleatorio del 1 al 10 y con ayuda del método de clase para la posición en la columna, asimismo asignándole con ayuda de los SET.

```
19          actTablero(ejercito_1, tablero);
20          actTablero(ejercito_2, tablero);
```

```
117     public static void actTablero(ArrayList<Soldado> ejercito, char[][] tablero){
118         for(int i = 0; i < ejercito.size(); i++){
119             int fila = ejercito.get(i).getFila();
120             char columna = ejercito.get(i).getColumna();
121             if(tablero[fila - 1][nroColumna(columna)] != '-'){
122                 fila = (int)(Math.random() * 10 + 1);
123                 columna = Soldado.numCol();
124             }
125             ejercito.get(i).setFila(fila);
126             ejercito.get(i).setCol(columna);
127             tablero[fila - 1][nroColumna(columna)] = ejercito.get(i).getFigura();
128         }
129     }
```

```
129     public static int nroColumna(char n){
130         switch(n){
131             case 'a': return 0;
132             case 'b': return 1;
133             case 'c': return 2;
134             case 'd': return 3;
135             case 'e': return 4;
136             case 'f': return 5;
137             case 'g': return 6;
138             case 'h': return 7;
139             case 'i': return 8;
140             case 'j': return 9;
141             default: return 0;
142         }
143     }
```

Ahora empezaremos con las opciones que podemos realizar:

```
25          boolean continuar = true;
26          while(continuar){
27              System.out.println("******************************");
28              System.out.println("Ingrese una opcion");
29              int opcion = sc.nextInt();
30              System.out.println("******************************");
31              switch(opcion){
32                  case 1:
33                      System.out.println("El tablero es ");
34                      impTablero(tablero);
35                      break;
36                  case 2:
37                      mayorVida(ejercito_1, 1);
38                      break;
39                  case 3:
40                      mayorVida(ejercito_2, 2);
41                      break;
42                  case 4:
43                      promVida(ejercito_1, 1);
44                      break;
45                  case 5:
46                      promVida(ejercito_2, 2);
47                      break;
48                  case 6:
49                      System.out.println("Los datos del ejercito 1 son : ");
50                      mostDatos(ejercito_1);
51                      break;
52                  case 7:
53                      System.out.println("Los datos del ejercito 2 son : ");
54                      mostDatos(ejercito_2);
55                      break;
```

```
56                  case 8:
57                      System.out.println("Primer metodo de orden - Ejercito 1");
58                      copiaDatos(ejercito_1, ejercito1_cop1);
59                      ordenarBurbuja(ejercito1_cop1);
60                      ejercito1_cop1.clear();
61                      break;
62                  case 9:
63                      System.out.println("2do metodo de orden - Ejercito 1");
64                      copiaDatos(ejercito_1, ejercito1_cop2);
65                      ordenarSeleccion(ejercito1_cop2);
66                      ejercito1_cop2.clear();
67                      break;
68                  case 10:
69                      System.out.println("1er metodo de orden - Ejercito 2");
70                      copiaDatos(ejercito_2, ejercito2_cop1);
71                      ordenarBurbuja(ejercito2_cop1);
72                      ejercito2_cop1.clear();
73                      break;
74                  case 11:
75                      System.out.println("2do metodo de orden - Ejercito 2");
76                      copiaDatos(ejercito_2, ejercito2_cop2);
77                      ordenarSeleccion(ejercito2_cop2);
78                      ejercito2_cop2.clear();
79                      break;
80                  case 12:
```

```
84                  case 13:
85                      System.out.println("Saliendo del programa");
86                      continuar = false;
87                      break;
88                  case 14:
89                      System.out.println("Sufrieron ataques del ejercito 1:");
90                      probAtacado(ejercito_1);
91                      break;
92                  case 15:
93                      System.out.println("Sufrieron ataques del ejercito 2:");
94                      probAtacado(ejercito_2);
95                      break;
96                  case 16:
97                      System.out.println("Rol del ejercito 1:");
98                      probRol(ejercito_1);
99                      System.out.println("Rol del ejercito 2:");
100                     probRol(ejercito_2);
101                     break;
102                 default:
103                     System.out.println("Opcion no valida");
104             }
105         }
```

Los break utilizados en cada case son usados porque los case solo revisan una línea en cambio con el break solo revisarían una sección y repetirían el ciclo.

Metodo mayorVida – Encargado de hacer una búsqueda lineal primero encontrando el punto mayor de vida, una vez realizado eso hace una búsqueda en todo el ArrayList si lo detecta este se imprimirá

```
153     public static void mayorVida(ArrayList<Soldado> ejercito, int n){
154         int mayor = ejercito.get(0).getVida();
155         for(int i = 1; i < ejercito.size(); i++){
156             if(ejercito.get(i).getVida() > mayor)
157                 mayor = ejercito.get(i).getVida();
158         }
159         System.out.println("Los soldados con mayor vida del ejercito " + n + " son:");
160         for(int j = 0; j < ejercito.size(); j++){
161             if(ejercito.get(j).getVida() == mayor){
162                 System.out.println(ejercito.get(j));
163             }
164         }
165     }
```

Metodo promVida – Hace una suma de todos los puntajes de vida y lo divide entre la cantidad de Soldado por ArrayList

```
166     public static void promVida(ArrayList<Soldado> ejercito, int m){
167         double n = 0;
168         for(int i = 0; i < ejercito.size(); i++)
169             n += ejercito.get(i).getVida();
170         n /= ejercito.size();
171         System.out.println("El promedio de vida del ejercito " + m  + " es " + n);
172     }
```

Metodo mostDatos – Encargado de imprimir todos los datos de los Soldado – ArrayList con ayuda del método de instancia toString de la Clase Soldado

```
210     public static void mostDatos(ArrayList<Soldado> ejercito){
211         for(Soldado n: ejercito)
212             System.out.println(n);
213     }
```

Metodo ganBatalla – El método que se uso para definir al ganador es al ejercito con mayor puntaje de vida.

```
239     public static void ganBatalla(ArrayList<Soldado> ejercito1, ArrayList<Soldado> ejercito2){
240         int suma1 = 0, suma2 = 0;
241         for(int i = 0; i < ejercito1.size(); i++){
242             suma1 += ejercito1.get(i).getVida();
243         }
244         for(int j = 0 ; j < ejercito2.size(); j++){
245             suma2 += ejercito2.get(j).getVida();
246         }
247         if(suma1 > suma2){
248             System.out.println("El ganador es el ejercito 1");
249         } else if( suma2 > suma1){
250             System.out.println("El ganador es el ejercito 2");
251         } else {
252             System.out.println("Es un empate");
253         }
254     }
```

Metodo probAtacado – Este método es nuevo ya que hara uso al método de instacia serAtacado de la clase Soldado dondre habrá una probabilida de 50 % si es atacado.

```
214     public static void probAtacado(ArrayList<Soldado> ejercito){
215         for(int i = 0; i < ejercito.size(); i++){
216             int n = (int)(Math.random() * 2);
217             if(n == 0){
218                 System.out.println("El soldado " + ejercito.get(i).getNombre() + " fue atacado");
219                 ejercito.get(i).serAtacado();
220                 System.out.println("Tiene de " + ejercito.get(i).getNivVidAct()  + " puntos de vida");
221             }
222         }
223     }
```

Metodo probRol – Este método mandara de forma aleatoria la actitud que tendrá el Soldado mediante condicionales.

```
224     public static void probRol(ArrayList<Soldado> ejercito){
225         for(int i = 0; i < ejercito.size(); i++){
226             int n = (int)(Math.random() * 3);
227             if(n == 0){
228                 System.out.println("El soldado " + ejercito.get(i).getNombre() + " atacara");
229                 ejercito.get(i).atacar();
230             } else if( n == 1){
231                 System.out.println("El soldado " + ejercito.get(i).getNombre() + " defendera");
232                 ejercito.get(i).defender();
233             } else{
234                 System.out.println("El soldado " + ejercito.get(i).getNombre() + " huira");
235                 ejercito.get(i).huir();
236             }
237         }
238     }
```

Metodos de Ordenamiento:

Para estos métodos como bien se escribió, hemos hecho un método que haga una copia de los valores del ArrayList Original a una copia. Necesario para no hacer una copia de la referencia del original y este no se modifique.

```
21          ArrayList<Soldado> ejercito1_cop1 = new ArrayList<Soldado>();
22          ArrayList<Soldado> ejercito1_cop2 = new ArrayList<Soldado>();
23          ArrayList<Soldado> ejercito2_cop1 = new ArrayList<Soldado>();
24          ArrayList<Soldado> ejercito2_cop2 = new ArrayList<Soldado>();
```

```
56                  case 8:
57                      System.out.println("Primer metodo de orden - Ejercito 1");
58                      copiaDatos(ejercito_1, ejercito1_cop1);
59                      ordenarBurbuja(ejercito1_cop1);
60                      ejercito1_cop1.clear();
61                      break;
62                  case 9:
63                      System.out.println("2do metodo de orden - Ejercito 1");
64                      copiaDatos(ejercito_1, ejercito1_cop2);
65                      ordenarSeleccion(ejercito1_cop2);
66                      ejercito1_cop2.clear();
67                      break;
68                  case 10:
69                      System.out.println("1er metodo de orden - Ejercito 2");
70                      copiaDatos(ejercito_2, ejercito2_cop1);
71                      ordenarBurbuja(ejercito2_cop1);
72                      ejercito2_cop1.clear();
73                      break;
74                  case 11:
75                      System.out.println("2do metodo de orden - Ejercito 2");
76                      copiaDatos(ejercito_2, ejercito2_cop2);
77                      ordenarSeleccion(ejercito2_cop2);
78                      ejercito2_cop2.clear();
79                      break;
```

En el método copia y los métodos de orden haremos uso del constructor sobrecargado de 11 parámetros.

```
173     public static void copiaDatos(ArrayList<Soldado> ejercito, ArrayList<Soldado> copia){
174         for(int i = 0; i < ejercito.size(); i++){
175             Soldado pos = ejercito.get(i);
176             Soldado c = new Soldado(pos.getNombre(), pos.getAtaque(), pos.getDefensa(), pos.getVida(), pos.getNivVidAct(), p
177             copia.add(c);
178         }
179     }
```

Metodo Burbuja:

```
195  public static void ordenarBurbuja(ArrayList<Soldado> ejercito){
196      for(int i = 0; i < ejercito.size() - 1; i++){
197          for(int j = 0; j < ejercito.size() - 1; j++){
198              Soldado may = ejercito.get(i);
199              Soldado men = ejercito.get(i + 1);
200              if(may.getVida() < men.getVida()){
201                  Soldado mayor = new Soldado(men.getNombre(), men.getAtaque(), men.getDefensa(), men.getVida(), men.getNivVi
202                  Soldado menor = new Soldado(may.getNombre(), may.getAtaque(), may.getDefensa(), may.getVida(), may.getNivVi
203                  ejercito.set(j, mayor);
204                  ejercito.set(j + 1, menor);
205              }
206          }
207      }
208      mostDatos(ejercito);
209  }
```

Metodo Selección:

```
180  public static void ordenarSeleccion(ArrayList<Soldado> ejercito){
181      for(int i = 0; i < ejercito.size() - 1; i++){
182          for(int j = i + 1; j < ejercito.size(); j++){
183              Soldado may = ejercito.get(i);
184              Soldado men = ejercito.get(j);
185              if(men.getVida() > may.getVida()){
186                  Soldado mayor = new Soldado(men.getNombre(), men.getAtaque(), men.getDefensa(), men.getVida(), men.getNivVi
187                  Soldado menor = new Soldado(may.getNombre(), may.getAtaque(), may.getDefensa(), may.getVida(), may.getNivVi
188                  ejercito.set(i, mayor);
189                  ejercito.set(j, menor);
190              }
191          }
192      }
193      mostDatos(ejercito);
194  }
```

PRUEBAS:

```
El ejercito 1 tiene un total de 4 soldados
El ejercito 2 tiene un total de 3 soldados
******************************
Ingrese una opcion
6
******************************
Los datos del ejercito 1 son :
Nombre: Soldado 1X1 Vida:1 Fila:10 Columna:i Actitud:Defensiva
Nombre: Soldado 2X1 Vida:5 Fila:6 Columna:b Actitud:Defensiva
Nombre: Soldado 3X1 Vida:3 Fila:7 Columna:h Actitud:Defensiva
Nombre: Soldado 4X1 Vida:2 Fila:10 Columna:g Actitud:Defensiva
******************************
Ingrese una opcion
7
******************************
Los datos del ejercito 2 son :
Nombre: Soldado 1X2 Vida:2 Fila:2 Columna:c Actitud:Defensiva
Nombre: Soldado 2X2 Vida:3 Fila:1 Columna:a Actitud:Defensiva
Nombre: Soldado 3X2 Vida:4 Fila:6 Columna:e Actitud:Defensiva
******************************
Ingrese una opcion
15
******************************
Rol del ejercito 1:
El soldado Soldado 1X1 huida
El soldado Soldado 2X1 defendera
El soldado Soldado 3X1 atacara
El soldado Soldado 4X1 huida
Rol del ejercito 2:
El soldado Soldado 1X2 atacara
El soldado Soldado 2X2 atacara
El soldado Soldado 3X2 atacara
******************************
```

```
******************************
Ingrese una opcion
6
******************************
Los datos del ejercito 1 son :
Nombre: Soldado 1X1 Vida:1 Fila:10 Columna:i Actitud:Fuga
Nombre: Soldado 2X1 Vida:5 Fila:6 Columna:b Actitud:Defensiva
Nombre: Soldado 3X1 Vida:3 Fila:7 Columna:h Actitud:Ofensiva
Nombre: Soldado 4X1 Vida:2 Fila:10 Columna:g Actitud:Fuga
******************************
Ingrese una opcion
7
******************************
Los datos del ejercito 2 son :
Nombre: Soldado 1X2 Vida:2 Fila:2 Columna:c Actitud:Ofensiva
Nombre: Soldado 2X2 Vida:3 Fila:1 Columna:a Actitud:Ofensiva
Nombre: Soldado 3X2 Vida:4 Fila:6 Columna:e Actitud:Ofensiva
******************************
Ingrese una opcion
13
******************************
Sufrieron ataques del ejercito 1:
El soldado Soldado 1X1 fue atacado
Tiene de 0 puntos de vida
El soldado Soldado 3X1 fue atacado
Tiene de 2 puntos de vida
******************************
Ingrese una opcion
14
******************************
Sufrieron ataques del ejercito 2:
El soldado Soldado 3X2 fue atacado
Tiene de 3 puntos de vida
```

```
*****************************
Los datos del ejercito 1 son :
Nombre: Soldado 1X1 Vida:0 Fila:10 Columna:i Actitud:Fuga
Nombre: Soldado 2X1 Vida:5 Fila:6 Columna:b Actitud:Defensiva
Nombre: Soldado 3X1 Vida:2 Fila:7 Columna:h Actitud:Ofensiva
Nombre: Soldado 4X1 Vida:2 Fila:10 Columna:g Actitud:Fuga
*****************************
Ingrese una opcion
7
*****************************
Los datos del ejercito 2 son :
Nombre: Soldado 1X2 Vida:2 Fila:2 Columna:c Actitud:Ofensiva
Nombre: Soldado 2X2 Vida:3 Fila:1 Columna:a Actitud:Ofensiva
Nombre: Soldado 3X2 Vida:3 Fila:6 Columna:e Actitud:Ofensiva
*****************************
Ingrese una opcion
11
*****************************
2do metodo de orden - Ejercito 2
Nombre: Soldado 3X2 Vida:3 Fila:6 Columna:e Actitud:Ofensiva
Nombre: Soldado 2X2 Vida:3 Fila:1 Columna:a Actitud:Ofensiva
Nombre: Soldado 1X2 Vida:2 Fila:2 Columna:c Actitud:Ofensiva
*****************************
Ingrese una opcion
9
*****************************
2do metodo de orden - Ejercito 1
Nombre: Soldado 2X1 Vida:5 Fila:6 Columna:b Actitud:Defensiva
Nombre: Soldado 3X1 Vida:2 Fila:7 Columna:h Actitud:Ofensiva
Nombre: Soldado 4X1 Vida:2 Fila:10 Columna:g Actitud:Fuga
Nombre: Soldado 1X1 Vida:0 Fila:10 Columna:i Actitud:Fuga
*****************************
Ingrese una opcion
16
*****************************
```