

Informe de Laboratorio 07

Tema: Combinando Arreglos Estandar y ArrayList

| Nota |
|------|
| |

| Estudiante | Escuela | Asignatura |
|---|--|--|
| Hernan Andy Choquehuanca Zapana hchoquehuanca@unsa.edu.pe | Escuela Profesional de Ingeniería de Sistemas | Fundamentos de la Programación II Semestre: II Código: 20232191 |

| Laboratorio | Tema | Duración |
|-------------|---|----------|
| 07 | Combinando Arreglos Estandar y ArrayList | 02 horas |

| Semestre académico | Fecha de inicio | Fecha de entrega |
|--------------------|---------------------|--------------------|
| 2023 - B | Del 18 Octubre 2023 | Al 23 Octubre 2023 |

1. Tarea

- Cree un Proyecto llamado Laboratorio7
- Usted deberá crear las dos clases Soldado.java y VideoJuego4.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.
- Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada.
- Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados y sus puntos de vida. Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Pro 22H2 64 bits.
- Visual Studio Code.
- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- Editor LaTeX en línea Overleaf.
- Variables Simples
- Métodos.
- Métodos de Búsqueda y Ordenamiento.
- Arreglos Estándar y Bidimensionales.
- ArrayList

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/hernanchoquehuanca/fp2-23b.git>
- URL para el laboratorio 07 en el Repositorio GitHub.
- <https://github.com/hernanchoquehuanca/fp2-23b/tree/main/fase02/lab07>

4. Trabajo del Laboratorio 07

4.1. Actividad 01

4.1.1. Clase Soldado.java

- Haciendo uso de la clase Soldado.java del laboratorio anterior (lab06), se adaptó al ejercicio actual.
 - Primero copiamos el código tal cual a la carpeta del laboratorio actual (lab07).

Listing 1: Commit 8809621: En el primer commit se reutilizaba la clase Soldado.java del laboratorio anterior (lab06)

```
$ git add .  
$ git commit -m "Reutilizando las clases Soldado.java y VideoJuego3.java del Lab06"  
$ git push -u origin main
```

- Nuestra clase Soldado tendrá los siguientes atributos:

- name (Nombre).
- row (Fila).
- column (Columna).
- status (Estado).
- health (Vida).
- team (Equipo).

```
2 private String name;  
3 private int row;  
4 private char column;  
5 private boolean status;  
6 private int health;  
7 private char team;
```

- Además creamos el método constructor, teniendo en cuenta que tiene que tener el mismo nombre que la clase y considerando sus atributos como parámetros:

- Se utilizó el puntero `this` para evitar ambigüedades.

```
9 public Soldado(String name, int row, char column, boolean status, int health, char team) {  
10     this.name = name;  
11     this.row = row;  
12     this.column = column;  
13     this.status = status;  
14     this.health = health;  
15     this.team = team;  
16 }
```

- Luego de ello también se implementaron los getters y setters para cada atributo de nuestra clase Soldado.

- Setters:

```
18 public void setName(String n){  
19     name = n;  
20 }  
21 public void setRow(int r){  
22     row = r;  
23 }  
24 public void setColumn(char c){  
25     column = c;  
26 }  
27 public void setStatus(boolean s){  
28     status = s;  
29 }  
30 public void setHealth(int h){  
31     health = h;  
32 }  
33 public void setTeam(char t){  
34     team = t;  
35 }
```

- Getters:

```
37 public String getName(){
38     return name;
39 }
40 public int getRow(){
41     return row;
42 }
43 public char getColumn(){
44     return column;
45 }
46 public boolean getStatus(){
47     return status;
48 }
49 public int getHealth(){
50     return health;
51 }
52 public char getTeam(){
53     return team;
54 }
```

Listing 2: Commit a43020e - 61b7b8e: Se concluía la clase Soldado.java agregando su atributo team, que en el lab06 no se tomaba en cuenta

```
$ git add .
$ git commit -m "Agregando el atributo team a la clase Soldado.java, esto servira
    para usarlo al momento de mostrar la tabla y sea mas funcional, ademas se adapto
    el codigo para trabajar con este atributo"
$ git push -u origin main
\end{lstlisting}
```

- Y finalizando con esta clase, se creo el método toString().
 - Se agregó la anotación `@Override` para indicar que se está reemplazando el método de su clase padre `Objeto`.
 - En este método se considera los atributos de la clase (name, row, column, status, health, team).
 - Se utilizaron saltos de con ayuda del `\n`.

```
55 @Override
56 public String toString() {
57     return "Data { " +
58         "\n Name:  " + name +
59         "\n Row:   " + row +
60         "\n Column: " + column +
61         "\n Status: " + status +
62         "\n Health: " + health +
63         "\n}\n";
64 }
```

- Un ejemplo de como se muestra en la siguiente imagen:

```
EJERCITO "A"
Data {
  Name:  Soldier0XA
  Row:   9
  Column: I
  Status: true
  Health: 4
}

Data {
  Name:  Soldier1XA
  Row:   9
  Column: F
  Status: true
  Health: 2
}

Data {
  Name:  Soldier2XA
  Row:   5
  Column: J
  Status: true
  Health: 2
}
```

Figura 1

4.1.2. Clase VideoJuego.java

- Primero se comenzó con la creación de la clase.
- Posterior a ello se crearon dos variables de clase:
 - Un ArrayList bidimensional, el cual contendrá a los soldados del ejército.
 - Además de un ArrayList unidimensional que nos servirá para contener a los soldados creados previamente, de esta manera será más fácil trabajar con ellos en los futuros métodos.

```
8 public class VideoJuego4 {
9   static ArrayList<ArrayList<Soldado>> army = new ArrayList<>();
10  static ArrayList<Soldado> army1DA = new ArrayList<>();
11  static ArrayList<Soldado> army1DB = new ArrayList<>();
```

Listing 3: Commit a43020e - a9e7076: En el segundo commit se modificaba las variables de la clase VideoJuego4.java para utilizar las más adecuadas, luego más adelante se optó por utilizar ArrayList

```
$ git add .
$ git commit -m "Cambiando la estructura de datos para el tablero, se utilizara un
    arreglo bidimensional de soldados, el cual sera una variable de la clase
    VideoJuego4.java"
$ git push -u origin main
```

4.1.3. Método para la creación de los ejército

- El método tiene como nombre `createArmy()`.
- Primero se define el número de soldados que contendrá cada ejército, haciendo uso de `Math.random`.
- Luego utilizando un doble bucle for, se inicializa el ArrayList bidimensional de soldados en `null`.
- Ahora se llama dos veces al método `createArmyTeam` para realizar la creación de los dos ejércitos a partir de los tamaños ya establecidos anteriormente y además el argumento tipo char para el identificador de cada equipo.

```
88 public static void createArmy(){
89     int numSoldiersA = (int) (Math.random() * 10) + 1;
90     int numSoldiersB = (int) (Math.random() * 10) + 1;
91
92     for (int i = 0; i < 10; i++){
93         army.add(new ArrayList<>());
94         for (int j = 0; j < 10; j++)
95             army.get(i).add(null);
96     }
97
98     army1DA = createArmyTeam(numSoldiersA, army1DA, 'A');
99     army1DB = createArmyTeam(numSoldiersB, army1DB, 'B');
100 }
```

Listing 4: Commit 598a62d: Se implementó el método `createArmy()`

```
$ git add .
$ git commit -m "Adaptando los metodos createArmy y CreateArmyTeam para que trabajen
ahora con los arreglos bidimensionales de soldados que es la estructura de datos
de nuestro tablero"
$ git push -u origin main
```

4.1.4. Método para la creación de ejércitos

- El método tiene como nombre `createArmyTeam()`.
- Recibe como parámetros un entero que es el número de soldados del ejército, el ArrayList de soldados a utilizar y un String que es el char que va al final del nombre de los soldados, esto último nos ayuda a identificarlos mejor.
- Utilizando un do while se crean posiciones aleatorias en el ArrayList bidimensional, tomando como condición que dicha posición sea distinta de `null`.
- Finalmente se crea el soldado, se almacena en ambos arreglos y retorna el ArrayList unidimensional con los soldados creados.

```
102 public static ArrayList <Soldado> createArmyTeam(int numSoldiers, ArrayList <Soldado>
    army1D, char t){
103     for (int i = 0; i < numSoldiers; i++){
104         int row, col;
105
106         do {
107             row = (int) (Math.random() * 9) + 1;
108             col = (int) (Math.random() * 9) + 1;
109         } while (army.get(row).get(col) != null);
110
111         Soldado s = new Soldado("Soldier" + i + "X" + t, row + 1, (char) (col + 'A'), true,
            (int) (Math.random() * 5) + 1, t);
112         army1D.add(i, s);
113         army.get(row).set(col, s);
114     }
115     return army1D;
116 }
```

Listing 5: Commit 598a62d - a9e7076: Se modificó el tipo de estructura para el tablero

```
$ git add .
$ git commit -m "Cambiando la estructura de datos del tablero principal, se volvio a
    utilizar ArrayList de ArrayList de soldados, esto para facilitar que el programa
    sea iterativo y la opcion de crear nuevos ejercitos sea optima"
$ git push -u origin main
```

4.1.5. Método para mostrar la tabla con el ejército

- El método tiene como nombre `showArmyTable()`.
- La funcionalidad es simple:
 - Se crea el String (`linesDown`), este será usado para la impresión de las líneas inferiores de cada fila de recuadros.
 - Primeramente se imprime la parte superior, donde se encuentran las letras que indica las columnas. Seguido de una línea que representa la parte superior de la tabla.
 - Se utilizó un bucle for con dos bucles en su interior, siendo su principal para las filas y los secundarios para las columnas.
 - En el primero, inicia imprimiendo los datos de los soldados si es que se encuentran, se evalúa si contiene un Soldado o `null`; en caso de ser así, imprimirá “ —”, en caso de contener a un soldado completará el cuadrado de la tabla incluyendo ejército (A o B), y su nombre simplificado (S + número generado).
 - Dentro del segundo for imprime el número de fila, se agregó una condicional que nos sirve para darle un toque de simetría a esta enumeración de fila, y así evitar que al momento de imprimir el 10 recorra un espacio. Además en caso de que en una posición se encuentre a un soldado, se colocará la vida de este.
 - Finalmente se imprime `linesDown` que completaría la línea inferior de cada fila.

```

118 public static void showArmyTable(ArrayList <ArrayList <Soldado>> army){
119     System.out.println("\n          TABLA CON LAS UBICACIONES DE LOS SOLDADOS CREADOS:
120         \n");
121     String linesDown = "
122         |-----|-----|-----|-----|-----|-----|-----|-----|-----|
123         System.out.println("          A          B          C          D          E          F          G          H          I
124         J\n"
125         + "
126         -----");
127
128     for (int r = 0; r < army.size(); r++){
129         System.out.print(" |");
130         for (int c = 0; c < army.get(r).size(); c++){
131             System.out.print(" " + (army.get(r).get(c) != null ? ("\'" +
132                 army.get(r).get(c).getTeam() + "\' "
133                 + "S" + army.get(r).get(c).getName().charAt(7) + " |") : " |"));
134
135             System.out.print("\n" + (r+1) + ((r != 9) ? " |" : " |"));
136             for (int c = 0; c < army.get(r).size(); c++){
137                 System.out.print(" " + (army.get(r).get(c) != null ? "HP: " +
138                     army.get(r).get(c).getHealth() : " ") + " |");
139
140             System.out.println("\n" + linesDown );
141         }
142     }
143     System.out.println();
144 }

```

- Un ejemplo de como se muestra en la siguiente imagen:

| TABLA CON LAS UBICACIONES DE LOS SOLDADOS CREADOS: | | | | | | | | | | |
|--|---|---|---|----------------|----------------|---|----------------|---|----------------|---|
| | A | B | C | D | E | F | G | H | I | J |
| 1 | | | | | | | | | | |
| 2 | | | | 'A'S3 HP: 3 | | | | | | |
| 3 | | | | | 'A'S0 HP: 2 | | 'B'S1 HP: 1 | | 'A'S1 HP: 3 | |
| 4 | | | | | | | 'B'S0 HP: 2 | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | 'B'S4 HP: 5 | |
| 7 | | | | | 'A'S2 HP: 5 | | | | | |
| 8 | | | | | 'A'S4 HP: 4 | | 'B'S2 HP: 4 | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | 'B'S3 HP: 1 | | | | | |

Figura 2

Listing 6: Commit 4b6ccf8: Se adaptó e implementó el método para mostrar la tabla con los soldados de ambos ejércitos incluyendo su vida

```
$ git add .  
$ git commit -m "Redefiniendo el metodo showTable para que ahora muestre el tablero  
con soldados incluyendo su vida, la cual esta representada por HP: n, siendo n  
el nivel de vida"  
$ git push -u origin main
```

4.1.6. Método para mostrar los datos de los soldados de un ejército

- El método tiene como nombre `showArmyData()`.
- Este usa un `for each` para recorrer el `ArrayList` unidimensional de `Soldado`, y luego mostrar sus datos con el `System.out.println`, que a su vez este sigue el formato que se estableció en el método `toString()` de la clase `Soldado.java`.
- La impresión en consola será la misma que la figura 1.

```
139 public static void showArmyData(ArrayList <Soldado> army1D, char t){  
140     System.out.println("EJERCITO \"" + t + "\"");  
141     for (Soldado s : army1D)  
142         System.out.println(s);  
143 }
```

Listing 7: Commit 95583b4: Se adaptó e implementó el método para mostrar los datos de los soldados de un ejército

```
$ git add .  
$ git commit -m "Redefiniendo el metodo showArmyData, el cual nos servira para  
mostrara los soldados de los ejercitos segun el orden de su creacion"  
$ git push -u origin main
```

4.1.7. Método para mostrar aquellos soldados con más vida de un ejército

- El método tiene como nombre `moreHealt()`.
- Este recibe el `ArrayList` con los soldados de un ejército, además de un `char` que indica el nombre del equipo.
- Primero recorre el `ArrayList` unidimensional de soldados haciendo uso de un bucle `for`, de esta manera obtendrá el máximo de vida del ejército, el cual será almacenado en un entero `maxHealth`.
- Finalmente imprimirá aquellos soldados que tengan la vida igual a `maxHealth`, con un `for` y un `if` que controlará aquello.
- En la impresión se incluye el nombre del ejército antes de mostrar a los soldados.

```

145 public static void moreHealth(ArrayList <Soldado> army1MH, char t){
146     int maxHealth = -1;
147     for(Soldado s : army1MH)
148         if (s.getHealth() > maxHealth)
149             maxHealth = s.getHealth();
150
151     System.out.println("Soldado(s) con mayor vida del Ejercito " + t + ": ");
152     for (Soldado s : army1MH)
153         if (s.getHealth() == maxHealth)
154             System.out.println("Nombre: " + s.getName() + " Vida: " + s.getHealth());
155     System.out.println();
156 }

```

- Un ejemplo de como se muestra en la siguiente imagen:

```

Soldado(s) con mayor vida del Ejercito A:
Nombre: Soldier4XA  Vida: 5
Nombre: Soldier6XA  Vida: 5

Soldado(s) con mayor vida del Ejercito B:
Nombre: Soldier2XB  Vida: 5
Nombre: Soldier3XB  Vida: 5

```

Figura 3

Listing 8: Commit ba455c4: Se agregó el método `moreHealth()` que imprimirá aquellos soldados que tengan la mayor vida dentro de su ejército

```

$ git add .
$ git commit -m "Redefiniendo el metodo moreHealth, el cual mostrara aquellos
soldados o soldado con mas vida dentro del ejercito que reciba como parametro"
$ git push -u origin main

```

4.1.8. Método para hallar la suma de vida en un ejército

- El método tiene como nombre `sumHealth()`.
- Se utiliza un entero inicializado en 0 para mientras que se recorre el `ArrayList` unidimensional de `Soldado` con un bucle `for each`, este entero (`sum`) va almacenando la vida de todos los soldados.
- Finalmente se retorna el entero `sum`.

```

162 public static int sumHealth(ArrayList <Soldado> army1DSH){
163     int sum = 0;
164     for (Soldado s : army1DSH)
165         sum += s.getHealth();
166     return sum;
167 }

```

- Un ejemplo de como se muestra en la siguiente imagen:

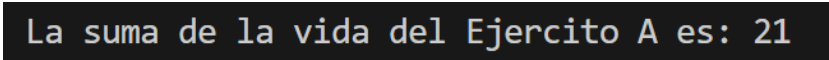


Figura 4

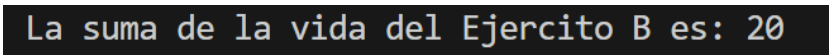


Figura 5

4.1.9. Método para hallar el promedio de vida en un ejército

- El método tiene como nombre `averageHealth()`.
- De manera breve como el método, este retorna una división entre la suma de la vida del ejército, haciendo uso del método `sumHealth()` y dividiendo entre el tamaño del ejército.

```
158 public static double averageHealth(ArrayList <Soldado> army1DAH){  
159     return sumHealth(army1DAH) / army1DAH.size();  
160 }
```

- Un ejemplo de como se muestra en la siguiente imagen:




Figura 6




Figura 7

Listing 9: Commit 422d59a: Se agregó el método `averageHealth` y `sumHealth`, siendo el segundo de utilidad para el primero

```
$ git add .  
$ git commit -m "Redefiniendo el metodo averageHealth, el cual calcula el promedio  
de vida de soldados en un ejercito, hace uso de un metodo llamado sumHealth que  
calcula la suma de la vida en un ejercito"  
$ git push -u origin main
```

4.1.10. Método de ordenamiento BubbleSort

- El método tiene como nombre `bubbleSort()`.
- El método tiene como finalidad ordenar el `ArrayList` unidimensional de `Soldado` haciendo uso del algoritmo `BubbleSort`, tomando en cuenta la vida de los soldados que contiene cada soldado de dicho `ArrayList`. Este algoritmo se extrajo de `Geeksforgeeks` y fue adaptado a este proyecto. **Extraído tal cual del laboratorio anterior (lab06)**

```
169 public static ArrayList <Soldado> bubbleSort(ArrayList <Soldado> army1DBS){
170     ArrayList <Soldado> army1DCopyBubble = new ArrayList<>(army1DBS);
171     int n = army1DCopyBubble.size();
172     boolean swapped;
173     for (int i = 0; i < n - 1; i++) {
174         swapped = false;
175         for (int j = 0; j < n - i - 1; j++)
176             if (army1DCopyBubble.get(j).getHealth() < army1DCopyBubble.get(j+1).getHealth()) {
177                 Soldado temp = army1DCopyBubble.get(j);
178                 army1DCopyBubble.set(j, army1DCopyBubble.get(j+1));
179                 army1DCopyBubble.set(j+1, temp);
180                 swapped = true;
181             }
182         if (!swapped)
183             break;
184     }
185     return army1DCopyBubble;
186 }
```

4.1.11. Método de ordenamiento InsertionSort

- El método tiene como nombre `insertionSort()`.
- El método tiene como finalidad ordenar el `ArrayList` unidimensional de `Soldado` haciendo uso del algoritmo `InsertionSort`, tomando en cuenta la vida de los soldados que contiene cada soldado de dicho `ArrayList`. Este algoritmo se extrajo de `Geeksforgeeks` y fue adaptado a este proyecto. **Extraído tal cual del laboratorio anterior (lab06)**

```
188 public static ArrayList <Soldado> insertionSort(ArrayList <Soldado> army1DIS) {
189     int n = army1DIS.size();
190     ArrayList <Soldado> army1DCopyInsertion = new ArrayList<>(army1DIS);
191
192     for (int i = 1; i < n; i++) {
193         Soldado key = army1DCopyInsertion.get(i);
194         int j = i - 1;
195
196         while (j >= 0 && army1DCopyInsertion.get(j).getHealth() < key.getHealth()) {
197             army1DCopyInsertion.set(j+1, army1DCopyInsertion.get(j));
198             j = j - 1;
199         }
200         army1DCopyInsertion.set(j+1, key);
201     }
202     return army1DCopyInsertion;
203 }
```

4.1.12. Método de ordenamiento SelectionSort

- El método tiene como nombre `selectionSort()`.
- El método tiene como finalidad ordenar el `ArrayList` unidimensional de `Soldado` haciendo uso del algoritmo `SelectionSort`, tomando en cuenta la vida de los soldados que contiene cada soldado de dicho `ArrayList`. Este algoritmo se extrajo de Geeksforgeeks y fue adaptado a este proyecto. Extraído tal cual del laboratorio anterior (lab06)

```
205 public static ArrayList <Soldado> selectionSort(ArrayList <Soldado> army1DSS) {
206     int n = army1DSS.size();
207     ArrayList <Soldado> army1DCopySelection = new ArrayList<>(army1DSS);
208
209     for (int i = 0; i < n - 1; i++) {
210         int min_idx = i;
211
212         for (int j = i + 1; j < n; j++)
213             if (army1DCopySelection.get(j).getHealth() >
214                 army1DCopySelection.get(min_idx).getHealth())
215                 min_idx = j;
216
217         Soldado temp = army1DCopySelection.get(min_idx);
218         army1DCopySelection.set(min_idx, army1DCopySelection.get(i));
219         army1DCopySelection.set(i, temp);
220     }
221     return army1DCopySelection;
222 }
```

4.1.13. Método para imprimir los soldados de un ejército, ordenados según su vida

- El método tiene como nombre `printArmyHealth()`.
- Este método recibirá un `ArrayList` unidimensional de `Soldado` (previamente ordenado), lo recorrerá usando un bucle `for` y mostrará los soldados, teniendo en cuenta que primero se mostrarán los de mayor vida hasta los de menor.

```
223 public static void printArmyHealth(ArrayList <Soldado> armyPrint, char t){
224     System.out.println("EJERCITO " + t + " : ");
225     for(int i = 0; i < armyPrint.size(); i++)
226         System.out.println((i + 1) + ". " + armyPrint.get(i).getName() + " Vida: " +
227             armyPrint.get(i).getHealth());
228 }
```

Listing 10: Commit fdb8f2c: Se implementará el método `printArmyHealth`

```
$ git add .
$ git commit -m "Redefiniendo el metodo printArmyHealth, el cual nos muestra los
    ArrayList de ejercitos ya ordenados segun el nivel de vida (mayor a menor)"
$ git push -u origin main
```

- Un ejemplo de como se muestra utilizando los 3 algoritmos de ordenamiento en la siguiente imagen:

```
Ejercitos ordenados (bubbleSort) segun la vida:
EJERCITO A:
1. Soldier0XA Vida: 4
2. Soldier1XA Vida: 1
EJERCITO B:
1. Soldier0XB Vida: 4
2. Soldier3XB Vida: 4
3. Soldier2XB Vida: 3
4. Soldier1XB Vida: 1

Ejercitos ordenados (insertionSort) segun la vida:
EJERCITO A:
1. Soldier0XA Vida: 4
2. Soldier1XA Vida: 1
EJERCITO B:
1. Soldier0XB Vida: 4
2. Soldier3XB Vida: 4
3. Soldier2XB Vida: 3
4. Soldier1XB Vida: 1

Ejercitos ordenados (selectionSort) segun la vida:
EJERCITO A:
1. Soldier0XA Vida: 4
2. Soldier1XA Vida: 1
EJERCITO B:
1. Soldier0XB Vida: 4
2. Soldier3XB Vida: 4
3. Soldier2XB Vida: 3
4. Soldier1XB Vida: 1
```

Figura 8

4.1.14. Método para mostrar al ejército ganador según la vida de sus soldados

- El método tiene como nombre `armyWinnerHealth()`.
- Este método utilizará la suma de vida de cada ejército, hará comparaciones y si uno de los dos es superior en número de vida, se imprimirá que ganó, en caso de ser iguales mostrará aquello.

```
229 public static void armyWinnerHealth(){
230     System.out.print("\nSegun la suma de vida de los soldados, ");
231     if (sumHealth(army1DA) > sumHealth(army1DB))
232         System.out.println("el ejercito ganador es el: \'A\'");
233     else if (sumHealth(army1DB) > sumHealth(army1DA))
234         System.out.println("el ejercito ganador es el: \'B\'");
235     else
236         System.out.println("la batalla quedo en empate");
237     System.out.println();
238 }
```

Listing 11: Commit e0d05ce: Se implementó el método armyWinnerHealth

```
$ git add .  
$ git commit -m "Redefiniendo el metodo armyWinnerHealth, el cual muestra un  
ejercito ganador o empate dependiendo de la suma de vida de los ejércitos"  
$ git push -u origin main
```

4.1.15. Método para mostrar la interfaz y volver el programa iterativo

- El método tiene como nombre `mainInterfaz()`.
- Al iniciar esta pequeña interfaz, se muestra las acciones que se pueden realizar, las cuales se eligen escribiendo un número del 1 - 8 según se desee.
- Al terminar cada caso desde el 1 al 7, se vuelve a llamar al mismo método, esto lo vuelve iterativo.
- En el caso 1, se hace el llamado al método `createArmy()`, esto para crear un nuevo ejército eliminando primero todos los ejércitos previamente creado (incluyendo el tablero).
- En el caso 2, se hace el llamado al método `showArmyData()`, esto mostrará los soldados de ambos ejércitos (A - B) siguiendo el orden de su creación.
- En el caso 3, se hace el llamado al método `showArmyTable()`, entonces mostrará el tablero con los soldados de ambos ejércitos, incluye su team, nombre y vida.
- En el caso 4, se hace el llamado al método `averageHealth()`, será realizado para ambos ejércitos, mostrando así el promedio de vida en cada uno.
- En el caso 5, se hace el llamado al método `moreHealth()`, aplicándolo a cada ejército mostrando así sus soldados con mayor cantidad de vida dentro de cada uno.
- En el caso 6, se hace el llamado al método `printArmyHealth()`, dentro de este se llama a 3 métodos por cada ejército los cuales son algoritmos de ordenamiento que ordenarán sus soldados de mayor a menor vida, para luego mostrarlos en consola.
- En el caso 7, se hace el llamado al método `armyWinnerHealth()`, mostrando así un ejército ganador en caso sus vidas no sean iguales, y en caso de que sí, mostrará aquello.
- En el caso 8, se muestra el mensaje **Fin** ya que esta opción termina el programa.
- Y por último en caso de no ser ningún número anteriormente mencionado, se muestra un mensaje indicando que se debe seleccionar una opción válida, y se hace un llamado nuevamente a la función.

```
16 public static void mainInterfaz(){
17     Scanner sc = new Scanner(System.in);
18     System.out.println("Bienvenido...");
19
20     System.out.println("1. Crear un nuevo ejercito"
21         + "\n2. Mostrar los datos de los ejercitos"
22         + "\n3. Mostrar la tabla con los ejercitos"
23         + "\n4. Mostrar el promedio de vida de los ejercitos"
24         + "\n5. Mostrar los soldados de ejercitos con mayor vida"
25         + "\n6. Ordenar los soldados de ejercitos segun la vida"
26         + "\n7. Mostrar ejercito ganador"
27         + "\n8. Salir del juego");
28     int action = sc.nextInt();
29
30     switch (action){
31         case 1 -> { // Crear un nuevo ejercito
32             army.clear();
33             army1DA.clear();
34             army1DB.clear();
35             createArmy();
36             mainInterfaz();
37         }
38         case 2 -> { // Mostrar los datos de los ejercitos
39             System.out.println("DATOS DE LOS SOLDADOS CREADOS:\n");
40             showArmyData(army1DA, 'A');
41             showArmyData(army1DB, 'B');
42             mainInterfaz();
43         }
44         case 3 -> { // Mostrar la tabla con los ejercitos
45             showArmyTable(army);
46             mainInterfaz();
47         }
48         case 4 -> { // Mostrar el promedio de vida de los ejercitos
49             System.out.println("El promedio de vida del Ejercito A es: " +
50                 averageHealth(army1DA));
51             System.out.println("El promedio de vida del Ejercito B es: " +
52                 averageHealth(army1DB));
53             mainInterfaz();
54         }
55         case 5 -> { // Mostrar los soldados de ejercitos con mayor vida
56             moreHelath(army1DA, 'A');
57             moreHelath(army1DB, 'B');
58             mainInterfaz();
59         }
60         case 6 -> { // Ordenar los soldados de ejercitos segun la vida
61             System.out.println("\nEjercitos ordenados (bubbleSort) segun la vida: ");
62             printArmyHealth(bubbleSort(army1DA), 'A');
63             printArmyHealth(bubbleSort(army1DB), 'B');
64
65             System.out.println("\nEjercitos ordenados (insertionSort) segun la vida: ");
66             printArmyHealth(insertionSort(army1DA), 'A');
67             printArmyHealth(insertionSort(army1DB), 'B');
68
69             System.out.println("\nEjercitos ordenados (selectionSort) segun la vida: ");
70             printArmyHealth(selectionSort(army1DA), 'A');
71             printArmyHealth(selectionSort(army1DB), 'B');
```



```
70     mainInterfaz();
71 }
72 case 7 -> { // Mostrar ejercito ganador
73     armyWinnerHealth();
74     mainInterfaz();
75 }
76 case 8 -> { // Salir del juego
77     System.out.println("Fin.");
78 }
79 default -> {
80     System.out.println("Selecciona una opcion valida");
81     mainInterfaz();
82 }
83 }
84
85 sc.close();
86 }
```

- Un ejemplo de como se muestra en la siguiente imagen:

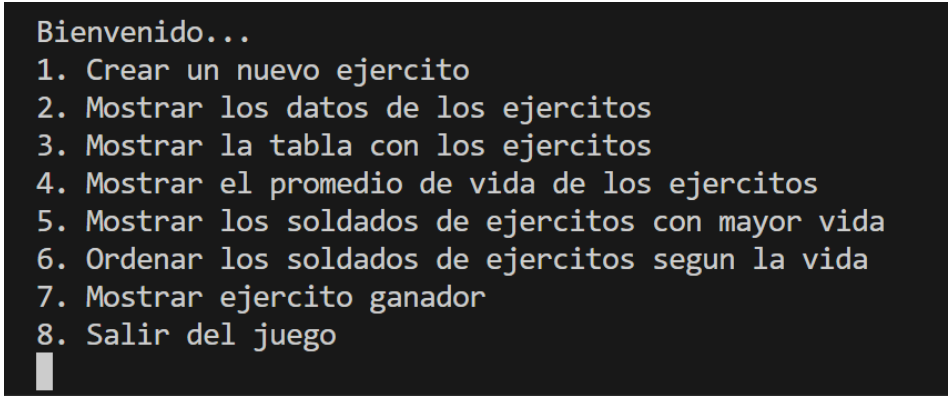


Figura 9

Listing 12: Commit 64fd210: Se implementó el método mainInterfaz (PRIMER COMMIT)

```
$ git add .
$ git commit -m "Agregando el metodo interfaz, el cual sera utilizado para hacer
nuestro programa iterativo"
$ git push -u origin main
```

Listing 13: Commit 390a95e: Se implementó el switch dentro del método mainInterfaz (SEGUNDO COMMIT)

```
$ git add .
$ git commit -m "Implementando en el metodo mainInterfaz un switch para recibir la
accion a realizar"
$ git push -u origin main
```

Listing 14: Commit 170f535: Se concluyó el switch dentro del método mainInterfaz tomando en cuenta todos los casos (ÚLTIMO COMMIT)

```
$ git add .  
$ git commit -m "Implementando el caso 7 donde se muestra un ejercito ganador o  
    empate, ya teniendo en cuenta la suma de vida de los soldados, esot dentro de la  
    funcion mainInterfaz"  
$ git push -u origin main
```

4.1.16. Método main, utilización de los métodos creados

- En el método principal (main) se utilizará simplemente dos métodos.
- Primero llamaremos al método `createArmy()` que inicializará los ejércitos, quiere decir sus Array-List y el tablero.
- Finalmente haciendo uso del método `mainInterfaz()` se inicia el programa, ya que llamará a este y se ejecutarán los métodos según el usuario solicite y acabará hasta que este lo decida.

```
12 public static void main(String [] args){  
13     createArmy();  
14     mainInterfaz();  
15 }
```

Listing 15: Commit 5410f63: Último commit, donde se implementó el método de la interfaz, sobre el cual está puesto los métodos del programa

```
$ git add .  
$ git commit -m "Terminando el codigo, se adapto el interfaz al metodo main y se  
    concluyo el metodo del mismo
```

4.2. Diagrama de clase UML



Figura 10

4.3. Estructura de laboratorio 07

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab07
| Soldado.java
| VideoJuego4.java
|
|-----latex
| Informe_Lab07.pdf
| Informe_Lab07.tex
|
|-----img
| averageHealth.png
| averageHealth2.png
| diagrama07.png
| logo_abet.png
| logo_episunsa.png
| logo_unsa.jpg
| mainInterfaz.png
| moreHealth.png
| printArmyHealth.png
| showArmyTable.png
| sumHealth.png
| sumHealth2.png
| toString.png
|
|-----src
| Soldado.java
| VideoJuego4.java
```

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 fácil 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 cumplió 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 auto calificarse 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 | 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 | |

6. Referencias

- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>
- <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>
- <https://docs.oracle.com/javase/tutorial/java/java00/methods.html>
- <https://www.geeksforgeeks.org/selection-sort/>
- <https://www.geeksforgeeks.org/bubble-sort/>
- <https://www.geeksforgeeks.org/insertion-sort/>
- <https://es.stackoverflow.com/questions/108171/>