Informe de Laboratorio 07

Tema: Combinando Arreglos Estándar y ArrayList

Nota

Estudiante	Escuela	Asignatura
Julio Rubén Chura Acabana jchuraaca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	F. de Programción 2 Semestre: I Código: 20230472

Laboratorio	Tema	Duración
07	Combinando Arreglos Estándar y ArrayList	04 horas

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

1. Tarea

- Tendrá 2 Ejércitos. 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 (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla).
- Realizar búsquedas secuencial y binaria en un ArrayList.
- Combinar arreglos estándar y ArrayList.
- Usted debe realizar varios commits y al término de la actividad deberá realizar un informe.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- Vim 9.0

- OpenJDK 64-Bits 20.0.2
- Git 2.42.0
- Cuenta en GitHub con el correo institucional
- ArrayList y Arreglos Estándar

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JulioChura/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/JulioChura/fp2-23b/tree/main/fase02/lab07`

4. Actividades con el repositorio GitHub

4.1. Inicialización del espacio de trabajo

Listing 1: Inicializando el espacio de trabajo

```
mkdir fase02
cd fase02
mkdir lab07
cd ..
cd fase01
cd lab06
Copy-Item "Soldier.java" -Destination "..\..\fase02\lab07"
Copy-Item "VideoJuego.java" -Destination "..\..\fase02\lab07\VideoJueg04.java"
cd ..
cd ..
cd fase02
cd lab07
vim VideoJuego4.java
```

Listing 2: Commit: 82bbf967de5142fdae59eac06dc2ed0cc640dd29

```
git add VideoJuego4.java
git commit -m "Se copia VideoJuego4.java del lab06 al lab07"
git push -u origin main
```

Listing 3: Commit: cf47e16c148de6bf8c57a1bb7d671ff4e5772537

```
git add VideoJuego4.java
git commit -m "Se copia la clase Soldier.java del lab06 al lab07"
git push -u origin main
```

4.2. Funciones ya trabajadas en laboratorios pasados

```java
public static ArrayList<ArrayList<Soldier>> generateArmy() {
    ArrayList<ArrayList<Soldier>> army = new ArrayList<ArrayList<Soldier>>(10);
    Random random = new Random();
    int amount = random.nextInt(10) + 1;
    int n = 0;
    for (int i = 0; i < 10; i++) {
        army.add(new ArrayList<>(Collections.nCopies(10, null)));
    }
    do {
        int row = random.nextInt(10);
        int column = random.nextInt(10);
        if (army.get(row).get(column) == null) {
            String name = "Soldier" + row + "x" + column;
            int lifePoints = random.nextInt(5) + 1;
            Soldier sol = new Soldier();
            sol.setLifePoints(lifePoints);
            sol.setName(name);
            sol.setColumn(column);
            sol.setRow(row);
            army.get(row).set(column, sol);
            n++;
        }
    } while (n < amount);
    return army;
}
```

- Método generateArmy(): Crea un arreglo bidimensional de 10 filas y 10 columnas con el fin de cubrir la misma cantidad de casillas del tablero, por lo que habrán posiciones que quedarán vacías ya que la cantidad de elementos del arreglo será un número aleatoria que oscila entre 1 a 10. También se genera el nombre de cada Soldier y sus puntos de vida de forma aleatoria. Este arreglo nos servirá más que nada para imprimir el tablero de una forma más sencilla.
- Como sabemos, el ArrayList es una estructura de datos compacta, es decir puede recibir n cantidad de elementos y esos elementos se acomodarán, por lo que no hay posiciones con elementos vacíos, pero es posible inicializar algunas posiciones con ningún elemento, en nuestro caso null, y eso hace el primer for, se encarga de llenar el ArrayList con null.

```
90      public static ArrayList<ArrayList<Soldier>> generateArmyB(ArrayList<ArrayList<Soldier>> a) {
91          ArrayList<ArrayList<Soldier>> army = new ArrayList<ArrayList<Soldier>>(10);
92          Random random = new Random();
93          int amount = random.nextInt(10) + 1;
94          int n = 0;
95
96          for (int i = 0; i < 10; i++) {
97              army.add(new ArrayList<>(Collections.nCopies(10, null)));
98          }
99          do {
100             int row = random.nextInt(10);
101             int column = random.nextInt(10);
102             if (army.get(row).get(column) == null && a.get(row).get(column) == null) {
103                 String name = "Soldier" + row + "x" + column;
104                 int lifePoints = random.nextInt(5) + 1;
105
106                 Soldier sol = new Soldier();
107
108                 sol.setLifePoints(lifePoints);
109                 sol.setName(name);
110                 sol.setColumn(column);
111                 sol.setRow(row);
112                 army.get(row).set(column, sol);
113                 n++;
114             }
115         } while (n < amount);
116         return army;
117     }
```

- Método generateArmyB:La razón de la creación de este método se debe a que hay casos en los que la cantidad de fichas del ejército B que están ubicadas en el tablero no coincide en cantidad con lo que se genera. Básicamente este método es igual que generateArmy, solo que este recibe un parámetro de tipo ArrayList bidimensional- Soldier que se usará para verificar que se generen los soldados del ejército B sin que haya cruces con los del ejército A. Lo único que se añade al método generateArmy es a.get(row).get(column) == null

```
52      public static ArrayList<Soldier> arrayListUnidimensional(ArrayList<ArrayList<Soldier>> s) {
53          ArrayList<Soldier> armyUni = new ArrayList<Soldier>();
54          for (int i = 0; i < s.size(); i++) {
55              for (int j = 0; j < s.get(i).size(); j++) {
56                  if (s.get(i).get(j) != null) {
57                      armyUni.add(s.get(i).get(j));
58                  }
59              }
60          }
61          return armyUni;
62      }
```

- arrayListUnidimensional: A pesar de ya contar con dos ArrayList bidimensionales, se opta por transformarlos en ArrayList unidimensionales ya que nos facilitará trabajar con los demás métodos que la práctica de laboratorio solicita. En el main se hace la creación de dos ArrayList unidimensionales tanto para el ejército A y B.

```
1    InsertionSort(arr[], n)
2        for i = 1 to n-1
3        key = arr[i]
4        j = i-1
5        while j >= 0 and arr[j] > key
6            arr[j+1] = arr[j]
7            j = j-1
8        arr[j+1] = key
```

```
153     public static Soldier longerLife(ArrayList<Soldier> s) {
154         int n = s.size();
155         for (int i = 1; i < n; i++) {
156             Soldier key = s.get(i);
157             int j = i - 1;
158             while (j >= 0 && s.get(j).getLifePoints() > key.getLifePoints()) {
159                 s.set(j + 1, s.get(j));
160                 j--;
161             }
162             s.set(j + 1, key);
163         }
164         return s.get(s.size() - 1);
165     }
```

- longerLife(): Se adapta el pseudocódigo de insertion y se le da la forma para que se aplique en un ArrayList. Lo que hace el método es ordenar de menor a mayor, por lo que último elemento es el mayor. El método devuelve al Soldier que se encuentra en esa posición.
- Se muestra el pseudocódigo, pero aplicado a un arreglo Estándar. A pesar de ser otra estructura, la idea es la misma

```
167     public static void showByCreation(ArrayList<Soldier> sol) {
168         for (Soldier n : sol) {
169             System.out.println(n);
170         }
171     }
```

- showByCreation(): Básicamente imprime los datos de un ArrayList unidimensional de tipo Soldier.

```
173     public static int totalLife(ArrayList<Soldier> sol) {
174         int addition = 0;
175         for (Soldier n : sol) {
176             addition = addition + n.getLifePoints();
177         }
178         return addition;
179     }
```

- totalLife(): El método devuelve los puntos de vida total del ejército. Se decide hacer el método con retorno para poder usar esos valores para determinar al ganador y calcular el promedio.

```java
    public static void orderByPower(ArrayList<Soldier> sol) {
        boolean swapped;
        Soldier temp;
        for (int i = 0; i < sol.size() - 1; i++) {
            swapped = false;
            for (int j = 0; j < sol.size() - 1 - i; j++) {
                temp = sol.get(j);
                sol.set(j, sol.get(j + 1));
                sol.set(j + 1, temp);
                swapped = true;
            }
            if (swapped == false) {
                break;
            }
        }
        for (Soldier n : sol) {
            System.out.println(n);
        }
    }
```

```
Procedimiento bubbleSort(entero arr[], entero n)
Inicio
    entero i, j
    booleano swapped
    Para i = 0 Hasta n - 1 Hacer
        swapped = falso
        Para j = 0 Hasta n - i - 1 Hacer
            Si arr[j] > arr[j + 1] Entonces
                intercambiar(arr[j], arr[j + 1])
                swapped = verdadero
            Fin Si
        Fin Para
        Si swapped == falso Entonces
            Romper
        Fin Si
    Fin Para
Fin
```

- orderByPower(): El método ordena el ArrayList tomando como criterio los puntos de Vida. No se retorna nada

- Se muestra el pseudocódigo, pero aplicado a un arreglo Estándar. A pesar de ser otra estructura, la idea es la misma.

```
203     public static void theWinner(int a, int b) {
204         if (a < b) {
205             System.out.println("El ganador es el equipo B");
206             System.out.println("Ventaja de " + (b - a) + " puntos de vida");
207         } else if (a > b) {
208             System.out.println("El ganador es el equipo A");
209             System.out.println("Ventaja de " + (a - b) + " puntos de vida");
210         } else {
211             System.out.println("Fue un empate");
212         }
213     }
```

- the Winner(): Este método recibe como parámetros dos enteros los cuales representan los puntos de vida total de cada ejército.Con ayuda de una estructura condicional se imprime un mensaje indicando al ganador o si hubo empate.

4.3. Desarrollo de nuevas funcionalidades

Listing 4: Se modifica los valores que el tablero mostrará

```
vim VideoJuego4.java
```

```
116     public static void myBoard(ArrayList<ArrayList<Soldier>> a, ArrayList<ArrayList<Soldier>> b) {
117         String[][] tablero = new String[10][10];
118         for (int i = 0; i < tablero.length; i++) {
119             for (int j = 0; j < tablero[i].length; j++) {
120                 tablero[i][j] = "|____|";
121             }
122         }
123         for (int i = 0; i < a.size(); i++) {
124             for (int j = 0; j < a.get(i).size(); j++) {
125                 if (a.get(i).get(j) != null) {
126                     String strA = "|_" + "a" + a.get(i).get(j).getLifePoints();
127                     tablero[i][j] = strA;
128                 }
129             }
130         }
131         for (int i = 0; i < b.size(); i++) {
132             for (int j = 0; j < b.get(i).size(); j++) {
133                 if (b.get(i).get(j) != null && tablero[i][j] != "s") {
134                     String strB = "|_" + "b" + b.get(i).get(j).getLifePoints();
135                     tablero[i][j] = strB;
136                 }
137             }
138         }
139         System.out.print("    A    B    C    D    E    F    G    H    I    J \n");
140         for (int i = 0; i < tablero.length; i++) {
141             System.out.printf("%2d", (i + 1));
142             for (int j = 0; j < tablero[i].length; j++) {
143                 System.out.print(tablero[i][j]);
144             }
145             System.out.println();
146         }
147     }
```

- En la práctica del laboratorio 7 se solicita que el tablero debe mostrar no solo las posiciones de los Soldier, sino también su vida. Para lograr ese objetivo se decide aumentar dos subguiones más a cada posición del arreglo tablero, eso con el fin de que haya una correcta distribución de los valores a mostrar (ver línea 120 de la imagen). Además se opta por concatenar los valores necesarios en una variable str de tipo String la cual será asignada en la posición del tablero correspondiente (en la imagen, ver las líneas 126, 127, 184 y 185 donde están los cambios).

Listing 5: Compilando y probando el tablero

```
javac VideoJuego4.java
java VideoJuego4
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier [name=Soldier6x6, lifePoints=1, row=7, column=7]
Soldier [name=Soldier7x8, lifePoints=4, row=8, column=9]
Soldier [name=Soldier9x0, lifePoints=5, row=10, column=1]
Mayor vida en A: Soldier [name=Soldier9x0, lifePoints=5, row=10, column=1]
El total de vida del ejercito A es: 10
El promedio de vida del ejercito A es: 3.3333333333333335
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier9x0, lifePoints=5, row=10, column=1]
Soldier [name=Soldier7x8, lifePoints=4, row=8, column=9]
Soldier [name=Soldier6x6, lifePoints=1, row=7, column=7]

DATOS DEL EJRCITO B
Soldier [name=Soldier9x4, lifePoints=1, row=10, column=5]
Mayor vida en B: Soldier [name=Soldier9x4, lifePoints=1, row=10, column=5]
El total de vida del ejercito B es: 1
El promedio de vida del ejercito B es: 1.0
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier9x4, lifePoints=1, row=10, column=5]

oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B     C     D     E     F     G     H     I     J
1 |____||____||____||____||____||____||____||____||____||____|
2 |____||____||____||____||____||____||____||____||____||____|
3 |____||____||____||____||____||____||____||____||____||____|
4 |____||____||____||____||____||____||____||____||____||____|
5 |____||____||____||____||____||____||____||____||____||____|
6 |____||____||____||____||____||____||____||____||____||____|
7 |____||____||____||____||____||____||_a1|____||____||____|
8 |____||____||____||____||____||____||____||____||_a4|____|
9 |____||____||____||____||____||____||____||____||____||____|
10|_a5|____||____||____||_b1|____||____||____||____||____|

++++++++++++++++++ FASE 3 DE LA CONTIENDA ++++++++++++++++++
El ganador se determina en base a los puntos de vida total
Enfrentamiento
El ganador es el equipo A
Ventaja de 9 puntos de vida
```

- En la ejecución del código nos damos cuenta que se está imprimiendo de forma correcta las posiciones, sin embargo, el formato no es el apropiado.

Listing 6: Commit: 44cebb1b28e0f5528952a1a7e67c71bd5f7a89b7

```
git add VideoJuego4.java
git commit -m "Primera version del tablero con la vida de cada Soldier incluida"
git push -u origin main
```

Listing 7: Se corrige el formato de impresión del tablero

```
vim VideoJuego4.java
```

```
116     public static void myBoard(ArrayList<ArrayList<Soldier>> a, ArrayList<ArrayList<Soldier>> b) {
117         String[][] tablero = new String[10][10];
118         for (int i = 0; i < tablero.length; i++) {
119             for (int j = 0; j < tablero[i].length; j++) {
120                 tablero[i][j] = "|____|";
121             }
122         }
123         for (int i = 0; i < a.size(); i++) {
124             for (int j = 0; j < a.get(i).size(); j++) {
125                 if (a.get(i).get(j) != null) {
126                     String strA = "|_" + "a" + a.get(i).get(j).getLifePoints() + "_|";
127                     tablero[i][j] = strA;
128                 }
129             }
130         }
131         for (int i = 0; i < b.size(); i++) {
132             for (int j = 0; j < b.get(i).size(); j++) {
133                 if (b.get(i).get(j) != null && tablero[i][j] != "s") {
134                     String strB = "|_" + "b" + b.get(i).get(j).getLifePoints() + "_|";
135                     tablero[i][j] = strB;
136                 }
137             }
138         }
139         System.out.print("   A     B     C     D     E    F     G     H     I     J \n");
140         for (int i = 0; i < tablero.length; i++) {
141             System.out.printf("%2d", (i + 1));
142             for (int j = 0; j < tablero[i].length; j++) {
143                 System.out.print(tablero[i][j]);
144             }
145             System.out.println();
146         }
147     }
```

- Se aumenta un subguión y la barra vertical tanto en la línea 126 y 134 para que se impriman las casillas donde se ubican los Soldier de forma armónica.

Listing 8: Compilando y probando el tablero

```
javac VideoJuego4.java
java VideoJuego4
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier [name=Soldier1x5, lifePoints=1, row=2, column=6]
Soldier [name=Soldier3x3, lifePoints=5, row=4, column=4]
```

```
Soldier [name=Soldier5x1, lifePoints=3, row=6, column=2]
Soldier [name=Soldier6x8, lifePoints=5, row=7, column=9]
Soldier [name=Soldier7x8, lifePoints=1, row=8, column=9]
Soldier [name=Soldier9x1, lifePoints=1, row=10, column=2]
Mayor vida en A: Soldier [name=Soldier6x8, lifePoints=5, row=7, column=9]
El total de vida del ejercito A es: 16
El promedio de vida del ejercito A es: 2.6666666666666665
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier6x8, lifePoints=5, row=7, column=9]
Soldier [name=Soldier3x3, lifePoints=5, row=4, column=4]
Soldier [name=Soldier5x1, lifePoints=3, row=6, column=2]
Soldier [name=Soldier9x1, lifePoints=1, row=10, column=2]
Soldier [name=Soldier7x8, lifePoints=1, row=8, column=9]
Soldier [name=Soldier1x5, lifePoints=1, row=2, column=6]

DATOS DEL EJRCITO B
Soldier [name=Soldier3x6, lifePoints=1, row=4, column=7]
Mayor vida en B: Soldier [name=Soldier3x6, lifePoints=1, row=4, column=7]
El total de vida del ejercito B es: 1
El promedio de vida del ejercito B es: 1.0
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier3x6, lifePoints=1, row=4, column=7]

ooooooooooooooo FASE 2 DE LA CONTIENDA ooooooooooooooo
Mostrando el tablero de juego
    A     B     C     D     E     F     G     H     I     J
1 |____||____||____||____||____||____||____||____||____||____|
2 |____||____||____||____||____||_a1_||____||____||____||____|
3 |____||____||____||____||____||____||____||____||____||____|
4 |____||____||____||_a5_||____||____||_b1_||____||____||____|
5 |____||____||____||____||____||____||____||____||____||____|
6 |____||_a3_||____||____||____||____||____||____||____||____|
7 |____||____||____||____||____||____||____||____||_a5_||____|
8 |____||____||____||____||____||____||____||____||_a1_||____|
9 |____||____||____||____||____||____||____||____||____||____|
10|____||_a1_||____||____||____||____||____||____||____||____|

++++++++++++++++++ FASE 3 DE LA CONTIENDA ++++++++++++++++++
El ganador se determina en base a los puntos de vida total
Enfrentamiento
El ganador es el equipo A
Ventaja de 15 puntos de vida
```

Listing 9: Commit: 8a5f07948da6ece26389a11919f8118d45bcea51

```
git add VideoJuego4.java
git commit -m "El tablero fue mejorado"
git push -u origin main
```

Listing 10: Se crea un método que valida la respuesta del jugador

```
vim VideoJuego4.java
```

```
212     public static boolean validation() {
213         Scanner sc = new Scanner(System.in);
214         do {
215             System.out.println("¿Desea jugar una ronda?(si/no)");
216             String answer = sc.next();
217             if (answer.equalsIgnoreCase("Si")) {
218                 return true;
219             } else if (answer.equalsIgnoreCase("No")) {
220                 return false;
221             } else {
222                 System.out.println("Respuesta no admsible");
223             }
224         } while (true);
225 
226     }
```

- Se crea este método para poder implementar más adelante un ciclo while en el main el cual permitirá que se juegue la cantidad de veces que el usuario deseé.
- En el método se usa el loop do while y una estructura condicional la que retornará un boolean de acuerdo a la respuesta del usuario y para que no hayan errores en las respuestas se usa el método equalsIgnoreCase(Str str) la cual ignora mayúsculas o minúsculas. Además de que en caso se ingrese una respuesta que no sea "si.º "no"se imprimirá el mensaje de Respuesta no admisible y se repetirá el ciclo hasta que se ingresen los valores correctos.

Listing 11: Commit: 47116e7456c1488e24e11bcd3518f2be3ec47bed

```
git add VideoJuego4.java
git commit -m "Se mejora el metodo que valida la respuesta del jugador"
git push -u origin main
```

Listing 12: Se implementa el loop While en el main para poder repetir una cantidad n de veces

```
vim VideoJuego4.java
```

```
 8 public class VideoJuego4 {
 9     public static void main(String[] args) {
10 
11         while (validation()) {
12             ArrayList<ArrayList<Soldier>> armyA = generateArmy();
13             ArrayList<ArrayList<Soldier>> armyB = generateArmyB(armyA);
14             ArrayList<Soldier> armyAU = arrayListUnidimensional(armyA);
15             ArrayList<Soldier> armyBU = arrayListUnidimensional(armyB);
```

- No hay grandes cambios en el main porque solo se mete todo lo que ya estaba en el loop (ver línea 11 de la imagen)
- En la condición del while se usa el método validation() para que el loop pueda funcionar de forma correcta sin causar un bucle infinito

Listing 13: Compilando y probando

```
javac VideoJuego4.java
java VideoJuego4
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier [name=Soldier0x4, lifePoints=2, row=1, column=5]
Soldier [name=Soldier0x8, lifePoints=1, row=1, column=9]
Soldier [name=Soldier0x9, lifePoints=4, row=1, column=10]
Soldier [name=Soldier1x7, lifePoints=2, row=2, column=8]
Soldier [name=Soldier2x1, lifePoints=3, row=3, column=2]
Soldier [name=Soldier2x7, lifePoints=4, row=3, column=8]
Soldier [name=Soldier3x9, lifePoints=5, row=4, column=10]
Soldier [name=Soldier5x8, lifePoints=1, row=6, column=9]
Soldier [name=Soldier6x3, lifePoints=3, row=7, column=4]
Soldier [name=Soldier8x9, lifePoints=4, row=9, column=10]
Mayor vida en A: Soldier [name=Soldier3x9, lifePoints=5, row=4, column=10]
El total de vida del ejercito A es: 29
El promedio de vida del ejercito A es: 2.9
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier3x9, lifePoints=5, row=4, column=10]
Soldier [name=Soldier8x9, lifePoints=4, row=9, column=10]
Soldier [name=Soldier2x7, lifePoints=4, row=3, column=8]
Soldier [name=Soldier0x9, lifePoints=4, row=1, column=10]
Soldier [name=Soldier6x3, lifePoints=3, row=7, column=4]
Soldier [name=Soldier2x1, lifePoints=3, row=3, column=2]
Soldier [name=Soldier1x7, lifePoints=2, row=2, column=8]
Soldier [name=Soldier0x4, lifePoints=2, row=1, column=5]
Soldier [name=Soldier5x8, lifePoints=1, row=6, column=9]
Soldier [name=Soldier0x8, lifePoints=1, row=1, column=9]

DATOS DEL EJRCITO B
Soldier [name=Soldier0x5, lifePoints=2, row=1, column=6]
Soldier [name=Soldier1x3, lifePoints=5, row=2, column=4]
Soldier [name=Soldier3x8, lifePoints=1, row=4, column=9]
Mayor vida en B: Soldier [name=Soldier1x3, lifePoints=5, row=2, column=4]
El total de vida del ejercito B es: 8
El promedio de vida del ejercito B es: 2.6666666666666665
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier1x3, lifePoints=5, row=2, column=4]
Soldier [name=Soldier0x5, lifePoints=2, row=1, column=6]
Soldier [name=Soldier3x8, lifePoints=1, row=4, column=9]

oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B     C     D     E     F     G     H     I     J
1 |____||____||____||____||_a2_||_b2_||____||____||_a1_||_a4_|
2 |____||____||____||_b5_||____||____||____||_a2_||____||____|
3 |____||_a3_||____||____||____||____||____||_a4_||____||____|
4 |____||____||____||____||____||____||____||____||_b1_||_a5_|
5 |____||____||____||____||____||____||____||____||____||____|
6 |____||____||____||____||____||____||____||____||_a1_||____|
```

```
    7 |____||____||____||_a3_||____||____||____||____||____||____|
    8 |____||____||____||____||____||____||____||____||____||____|
    9 |____||____||____||____||____||____||____||____||____||_a4_|
    10|____||____||____||____||____||____||____||____||____||____|

    ++++++++++++++++++ FASE 3 DE LA CONTIENDA ++++++++++++++++++
    El ganador se determina en base a los puntos de vida total
    Enfrentamiento
    El ganador es el equipo A
    Ventaja de 21 puntos de vida
    Desea jugar una ronda?(si/no)
    7
    Respuesta no admsible
    Desea jugar una ronda?(si/no)
    no
```

Listing 14: Commit: 41ffd9ca2d4934c86227d650940d5a93865bc037

```
    git add VideoJuego3.java
    git commit -m "Se crea el loop que permite jugar n cantidad de veces"
    git push -u origin main
```

Listing 15: Se implementa un método que realiza una búsqueda Binaria según el nombre

```
    vim VideoJuego4.java
```

```
function busqueda_binaria (datos, valor_busqueda):
    inicio = 0
    fin = longitud (datos) - 1
    mientras inicio <= fin:
        medio = (inicio + fin) // 2
        si datos [medio] == valor_busqueda:
            devolver medio
        sino si datos [medio] < valor_busqueda:
            inicio = medio + 1
        sino:
            fin = medio - 1
    devolver -1
```

- Este es el pseudocódigo general de la búsqueda binaria.

```java
    public static void binarySearchByName(ArrayList<Soldier> armyA, String name) {
        Collections.sort(armyA, new Comparator<Soldier>() {
            public int compare(Soldier soldier1, Soldier soldier2) {
                return soldier1.getName().compareTo(soldier2.getName());
            }
        });
        int high = armyA.size() - 1;
        int low = 0;
        while (low <= high) {
            int mid = (high + low) / 2;
            Soldier soldier = armyA.get(mid);
            if (name.equalsIgnoreCase(soldier.getName())) {
                System.out.println("Se ha encontrado: " + soldier);
                return;
            } else if (name.compareTo(soldier.getName()) < 0) {
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        System.out.println("No fue encontrado");
    }
```

- Para una mejor presentación del código, se decide por solo limitarse a mostrar mensajes al usuario y colocar los métodos en el main, esa es la razón por la que se decide hacer este método sin retorno.
- Pese a contar con un método de ordenamiento, ese es de tipo void, así que para ordenar este arreglo y hacer la búsqueda binaria, se tendrá que hacerlo en el mismo método, por lo que se hará uso del método Collections.sort del cual se hará una personalización en el parámetro de Comparator haciendo que el criterio sean los nombres de cada Soldier.

Listing 16: Compilando y probando

```
javac VideoJuego4.java
java VideoJuego4
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier [name=Soldier0x1, lifePoints=3, row=1, column=2]
Soldier [name=Soldier1x4, lifePoints=4, row=2, column=5]
Soldier [name=Soldier2x9, lifePoints=1, row=3, column=10]
Soldier [name=Soldier5x6, lifePoints=3, row=6, column=7]
Soldier [name=Soldier7x4, lifePoints=5, row=8, column=5]
Soldier [name=Soldier9x0, lifePoints=5, row=10, column=1]
Soldier [name=Soldier9x4, lifePoints=5, row=10, column=5]
Mayor vida en A: Soldier [name=Soldier9x4, lifePoints=5, row=10, column=5]
El total de vida del ejercito A es: 26
El promedio de vida del ejercito A es: 3.7142857142857144
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier9x4, lifePoints=5, row=10, column=5]
Soldier [name=Soldier9x0, lifePoints=5, row=10, column=1]
Soldier [name=Soldier7x4, lifePoints=5, row=8, column=5]
```

```
Soldier [name=Soldier1x4, lifePoints=4, row=2, column=5]
Soldier [name=Soldier5x6, lifePoints=3, row=6, column=7]
Soldier [name=Soldier0x1, lifePoints=3, row=1, column=2]
Soldier [name=Soldier2x9, lifePoints=1, row=3, column=10]
Ingrese el nombre del Soldier que desea buscar
Soldier0x1
Se ha encontrado: Soldier [name=Soldier0x1, lifePoints=3, row=1, column=2]

DATOS DEL EJRCITO B
Soldier [name=Soldier6x3, lifePoints=5, row=7, column=4]
Mayor vida en B: Soldier [name=Soldier6x3, lifePoints=5, row=7, column=4]
El total de vida del ejercito B es: 5
El promedio de vida del ejercito B es: 5.0
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier6x3, lifePoints=5, row=7, column=4]
Ingrese el nombre del Soldier que desea buscar
oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B     C     D     E     F     G     H     I     J
1 |____||_a3_||____||____||____||____||____||____||____||____|
2 |____||____||____||____||_a4_||____||____||____||____||____|
3 |____||____||____||____||____||____||____||____||____||_a1_|
4 |____||____||____||____||____||____||____||____||____||____|
5 |____||____||____||____||____||____||____||____||____||____|
6 |____||____||____||____||____||____||_a3_||____||____||____|
7 |____||____||____||_b5_||____||____||____||____||____||____|
8 |____||____||____||____||_a5_||____||____||____||____||____|
9 |____||____||____||____||____||____||____||____||____||____|
10|_a5_||____||____||____||_a5_||____||____||____||____||____|

+++++++++++++++++ FASE 3 DE LA CONTIENDA +++++++++++++++++
El ganador se determina en base a los puntos de vida total
Enfrentamiento
El ganador es el equipo A
Ventaja de 21 puntos de vida
Desea jugar una ronda?(si/no)
NO
```

Listing 17: Commit: dde0af374ba825d7f45cb32605658c8624920f51

```
git add VideoJuego4.java
git commit -m "Se culmina el metodo binarySearcByName"
git push -u origin main
```

Listing 18: Se implementa un método que realiza una búsqueda secuencial según el nombre

```
vim VideoJuego4.java
```

```java
    public static void sequenceSearchByName(ArrayList<Soldier> armyB, String name) {
        for (int i = 0; i < armyB.size(); i++) {
            if (name.equalsIgnoreCase(armyB.get(i).getName())) {
                System.out.println("Se ha encontrado: " + armyB.get(i));
                return;
            }
        }
        System.out.println("No se ha encontrado coincidencias");
    }
```

- Este método busca a lo largo del ArrayList elemento por elemento. La búsqueda secuencial no es una buena opción debido a que hace recorridos demás. Se lo emplea más que nada porque los objetivos de la práctica lo exigen.

Listing 19: Compilando y probando el código terminado

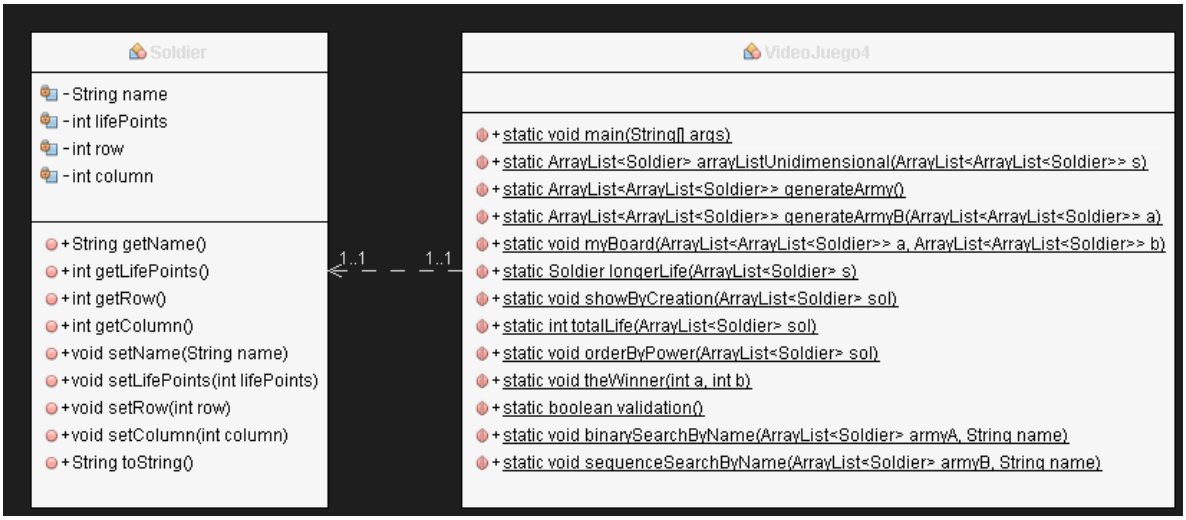
```
javac VideoJuego4.java
java VideoJuego4
Desea jugar una ronda?(si/no)
SI
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier [name=Soldier5x2, lifePoints=2, row=6, column=3]
Soldier [name=Soldier9x8, lifePoints=1, row=10, column=9]
Mayor vida en A: Soldier [name=Soldier5x2, lifePoints=2, row=6, column=3]
El total de vida del ejercito A es: 3
El promedio de vida del ejercito A es: 1.5
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier5x2, lifePoints=2, row=6, column=3]
Soldier [name=Soldier9x8, lifePoints=1, row=10, column=9]
Ingrese el nombre del Soldier que desea buscar
soldier5x2
Se ha encontrado: Soldier [name=Soldier5x2, lifePoints=2, row=6, column=3]

DATOS DEL EJRCITO B
Soldier [name=Soldier2x9, lifePoints=5, row=3, column=10]
Soldier [name=Soldier3x1, lifePoints=3, row=4, column=2]
Soldier [name=Soldier3x5, lifePoints=2, row=4, column=6]
Soldier [name=Soldier3x7, lifePoints=3, row=4, column=8]
Soldier [name=Soldier4x2, lifePoints=1, row=5, column=3]
Soldier [name=Soldier5x0, lifePoints=1, row=6, column=1]
Soldier [name=Soldier5x8, lifePoints=1, row=6, column=9]
Soldier [name=Soldier7x4, lifePoints=1, row=8, column=5]
Mayor vida en B: Soldier [name=Soldier2x9, lifePoints=5, row=3, column=10]
El total de vida del ejercito B es: 17
El promedio de vida del ejercito B es: 2.125
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier2x9, lifePoints=5, row=3, column=10]
Soldier [name=Soldier3x7, lifePoints=3, row=4, column=8]
Soldier [name=Soldier3x1, lifePoints=3, row=4, column=2]
Soldier [name=Soldier3x5, lifePoints=2, row=4, column=6]
Soldier [name=Soldier7x4, lifePoints=1, row=8, column=5]
Soldier [name=Soldier5x8, lifePoints=1, row=6, column=9]
Soldier [name=Soldier5x0, lifePoints=1, row=6, column=1]
```

```
Soldier [name=Soldier4x2, lifePoints=1, row=5, column=3]
Ingrese el nombre del Soldier que desea buscar
soldier4x2
Se ha encontrado: Soldier [name=Soldier4x2, lifePoints=1, row=5, column=3]

ooooooooooooooo FASE 2 DE LA CONTIENDA ooooooooooooooo
Mostrando el tablero de juego
    A     B     C     D     E     F     G     H     I     J
1 |____||____||____||____||____||____||____||____||____||____|
2 |____||____||____||____||____||____||____||____||____||____|
3 |____||____||____||____||____||____||____||____||____||_b5_|
4 |____||_b3_||____||____||____||_b2_||____||_b3_||____||____|
5 |____||____||_b1_||____||____||____||____||____||____||____|
6 |_b1_||____||_a2_||____||____||____||____||____||_b1_||____|
7 |____||____||____||____||____||____||____||____||____||____|
8 |____||____||____||____||_b1_||____||____||____||____||____|
9 |____||____||____||____||____||____||____||____||____||____|
10|____||____||____||____||____||____||____||____||_a1_||____|

+++++++++++++++++ FASE 3 DE LA CONTIENDA +++++++++++++++++
El ganador se determina en base a los puntos de vida total
Enfrentamiento
El ganador es el equipo B
Ventaja de 14 puntos de vida
Desea jugar una ronda?(si/no)
no
```

Listing 20: Commit: 5575a0d5eead5b5dd4dca7425a5b9f37229e25fa

```
git add VideoJuego4.java
git commit -m "Se culima el metood sequenceSearchByName"
git push -u origin main
```

4.4. Diagrama UML y el main



```java
public class VideoJuego4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (validation()) {
            ArrayList<ArrayList<Soldier>> armyA = generateArmy();
            ArrayList<ArrayList<Soldier>> armyB = generateArmyB(armyA);
            ArrayList<Soldier> armyAU = arrayListUnidimensional(armyA);
            ArrayList<Soldier> armyBU = arrayListUnidimensional(armyB);
            int a = totalLife(armyAU);
            int b = totalLife(armyBU);

            System.out.println("ooooooooooooooo  FASE 1 DE LA CONTIENDA  ooooooooooooooo");
            System.out.println("Mostrando estadisticas de cada ejercito" + "\n");
            System.out.println("Mostrando soldados por orden de creacion");
            System.out.println("DATOS DEL DEL EJERCITO A");
            showByCreation(armyAU);
            System.out.println("Mayor vida en A: " + longerLife(armyAU));
            System.out.println("El total de vida del ejercito A es: " + totalLife(armyAU));
            System.out.println("El promedio de vida del ejercito A es: " + (double) a / armyAU.size());
            System.out.println("Mostrando soldados por ranking de poder de A");
            orderByPower(armyAU);
            System.out.println("Ingrese el nombre del Soldier que desea buscar");
            String nameA = sc.next();
            binarySearchByName(armyAU, nameA);
            System.out.println();
            System.out.println("DATOS DEL EJRCITO B");
            showByCreation(armyBU);
            System.out.println("Mayor vida en B: " + longerLife(armyBU));
            System.out.println("El total de vida del ejercito B es: " + totalLife(armyBU));
            System.out.println("El promedio de vida del ejercito B es: " + (double) b / armyBU.size());
            System.out.println("Mostrando soldados por ranking de poder de B");
            orderByPower(armyBU);
            System.out.println("Ingrese el nombre del Soldier que desea buscar");
            String nameB = sc.next();
            sequenceSearchByName(armyBU, nameB);
            System.out.println();

            System.out.println("ooooooooooooooo  FASE 2 DE LA CONTIENDA  ooooooooooooooo");
            System.out.println("Mostrando el tablero de juego");
            myBoard(armyA, armyB);
            System.out.println();

            System.out.println("+++++++++++++++++   FASE 3 DE LA CONTIENDA  +++++++++++++++++");
            System.out.println("El ganador se determina en base a los puntos de vida total");
            System.out.println("Enfrentamiento");
            theWinner(a, b);
        }

    }
```

4.5. Estructura de laboratorio 07

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

```
lab08
|    Soldier.java
|
|    VideoJuego4.java
|
 ----latex
      |     programacion_lab07_rescobedoq_v1.0.pdf
      |     programacion_lab07_rescobedoq_v1.0.tex
      |
      |----img
      |         arrayListUni.jpg
      |         binarySearch.jpg
      |         board.jpg
      |         boardComplete.jpg
      |         burbuja.jpg
```

```
|        generateArmy.jpg
|        generateArmyB.jpg
|        insertion.jpg
|        logo_abet.png
|        logo_episunsa.png
|        logo_unsa.jpg
|        longerLife.png
|        loop.jpg
|        main.jpg
|        orderByPower.png
|        sequence.jpg
|        showByCreation.png
|        theWinner.jpg
|        totalLife.png
|        uml.png
|        validation.jpg
|
 ------src
        binary.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 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 cumplio 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 lo items.
- 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	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 estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1	
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		18	

6. Referencias

- https://www.geeksforgeeks.org/bubble-sort/
- https://www.geeksforgeeks.org/insertion-sort/