Informe de Laboratorio 06
Tema: 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
06	ArrayList	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 11 Octubre 2023	Al 16 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 Marco Aedo López 2 algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla).
- Usted debe realizar varios commits y al término de la actividad deberá realizar un informe.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- Notepad++ v8.5.4.
- OpenJDK 64-Bits 20.0.2.
- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- ArrayList

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/fase01/lab06`

4. Actividades con el repositorio GitHub

4.1. Desarrollo de la práctica de laboratorio

Listing 1: Inicializando el espacio de trabajo

```
mkdir lab06
cd lab05
Copy-Item "Soldier.java" -Destination "..\lab06"
Copy-Item "VideoJuego.java" -Destination "..\lab06\VideoJueg03.java"
cd ..
cd lab05
notepad++ Soldier.jav
```

Listing 2: Commit: Copiando los archivos java del lab05

```
git add VideoJuego3.java
git commit -m "Copiando los archivos java del lab05"
git push -u origin main
```

Listing 3: Se implementa un método que genera un ArrayList bidimensional de Soldier

```
Notepad++ Ejercicio02.java
```

```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;
}
```

- El 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

Listing 4: Commit: Metodo que genera un ejercito

```
git add VideoJuego3.java
git commit -m "Metodo que genera un ejercito"
git push -u origin main
```

Listing 5: Se implementa el método que imprime el tablero con los soldados ya ubicados

```
notepad++ VideoJuego3.java
```

```java
public static void myBoard(ArrayList<ArrayList<Soldier>> a, ArrayList<ArrayList<Soldier>> b) {
    String[][] tablero = new String[10][10];
    for (int i = 0; i < tablero.length; i++) {
        for (int j = 0; j < tablero[i].length; j++) {
            tablero[i][j] = "|___|";
        }
    }

    for (int i = 0; i < a.size(); i++) {
        for (int j = 0; j < a.get(i).size(); j++) {
            if (a.get(i).get(j) != null) {
                tablero[i][j] = "|_a_|";
            }
        }
    }

    for (int i = 0; i < b.size(); i++) {
        for (int j = 0; j < b.get(i).size(); j++) {
            if (b.get(i).get(j) != null && tablero[i][j] != "s") {
                tablero[i][j] = "|_b_|";
            }
        }
    }

    System.out.print("    A    B    C    D    E    F    G    H    I    J \n");
    for (int i = 0; i < tablero.length; i++) {
        System.out.printf("%2d", (i + 1));
        for (int j = 0; j < tablero[i].length; j++) {
            System.out.print(tablero[i][j]);
        }
        System.out.println();
    }
}
```

- Este método recibe como parámetros dos ArrayList bidimensionales de tipo Soldier. Se hace uso de 4 ciclos for. El primer for nos permite generar un arreglo bidimensional de caracteres que simularán el tablero. El segundo for se encarga de posicionar a los soldados del ejército A. El tercer for se encarga de posicionar a los soldados del ejército B. El último for se encarga imprimir el tablero con las posiciones y entradas correspondientes.

Listing 6: Compilando y probando el tablero

```
javac VideoJuego3.java
java VideoJuego3
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B    C    D    E    F    G    H    I    J
1 |___||___||___||___||___||___||___||___||___||___|
2 |___||___||___||___||_b_||___||___||___||___||___|
3 |___||___||___||___||_b_||___||___||___||___||___|
4 |___||_b_||___||___||___||___||___||___||___||_a_|
5 |___||___||___||___||___||___||___||___||___||___|
6 |___||___||___||___||___||___||___||___||___||___|
7 |___||___||___||___||___||___||___||___||___||___|
8 |___||___||___||___||___||___||___||___||___||___|
9 |___||___||___||_b_||___||___||___||___||___||___|
10|___||___||___||___||___||___||___||___||___||___|
```

Listing 7: Commit: Metodo que imprime el tablero completo

```
git add VideoJuego3.java
git commit -m "Metodo que imprime el tablero completo"
git push -u origin main
```

Listing 8: Se implementa el método que genera un ArrayList unidimensional

```
notepad++ VideoJuego3.java
```

```
42      public static ArrayList<Soldier> arrayListUnidimensional(ArrayList<ArrayList<Soldier>> s) {
43          ArrayList<Soldier> armyUni = new ArrayList<Soldier>();
44          for (int i = 0; i < s.size(); i++) {
45              for (int j = 0; j < s.get(i).size(); j++) {
46                  if (s.get(i).get(j) != null) {
47                      armyUni.add(s.get(i).get(j));
48                  }
49              }
50          }
51          return armyUni;
52      }
```

- 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.

Listing 9: Commit: Metodo que convierte un ArrayList Bidimensional a Unidimensional

```
git add VideoJuego3.java
git commit -m "Metodo que convierte un ArrayList Bidimensional a Unidimensional"
git push -u origin main
```

Listing 10: Se implementa el método que muestra los datos del ArrayList unidimensional según su orden de creación

```
notepad++ VideoJuego3.java
```

```
158     public static void showByCreation(ArrayList<Soldier> sol) {
159         for (Soldier n : sol) {
160             System.out.println(n);
161         }
162     }
```

- Para tener una claridad de lo que se está haciendo, recién se decide implementar este método para ver si se está imprimiendo correctamente las posiciones de los soldados en el tablero.

Listing 11: Compilando y probando

```
javac VideoJuego3.java
java VideoJuego3
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=Soldier0x6, lifePoints=3, row=1, column=7]
Soldier [name=Soldier1x7, lifePoints=3, row=2, column=8]
Soldier [name=Soldier3x6, lifePoints=4, row=4, column=7]
Soldier [name=Soldier4x1, lifePoints=1, row=5, column=2]
Soldier [name=Soldier4x2, lifePoints=1, row=5, column=3]
Soldier [name=Soldier5x0, lifePoints=3, row=6, column=1]
Soldier [name=Soldier5x3, lifePoints=3, row=6, column=4]
Soldier [name=Soldier7x9, lifePoints=3, row=8, column=10]
Soldier [name=Soldier8x4, lifePoints=5, row=9, column=5]

DATOS DEL EJERCITO B
Soldier [name=Soldier0x0, lifePoints=2, row=1, column=1]
Soldier [name=Soldier8x5, lifePoints=1, row=9, column=6]
Soldier [name=Soldier8x8, lifePoints=4, row=9, column=9]
oooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooo
Mostrando el tablero de juego
   A     B    C    D    E    F    G    H    I    J
1 |_b_||___||___||___||___||___||_a_||___||___||___|
2 |___||___||___||___||___||___||___||_a_||___||___|
3 |___||___||___||___||___||___||___||___||___||___|
4 |___||___||___||___||___||___||_a_||___||___||___|
5 |___||_a_||_a_||___||___||___||___||___||___||___|
6 |_a_||___||___||_a_||___||___||___||___||___||___|
7 |___||___||___||___||___||___||___||___||___||___|
8 |___||___||___||___||___||___||___||___||___||_a_|
9 |___||___||___||___||_a_||_b_||___||___||_b_||___|
10|___||___||___||___||___||___||___||___||___||___|
```

- Si observamos, nos damos cuenta que en el tablero falta generar la posición de un soldado del ejército B. Esto nos hace indicar que no se está haciendo correctamente el reemplazo para generar el tablero.

Listing 12: Commit: Método que muestra los datos del ArrayList

```
git add VideoJuego3.java
git commit -m "Metodo que muestra los datos del ArrayList"
git push -u origin main
```

Listing 13: Se implementa un método que genera un ArrayList para el ejército B

```
notepad++ VideoJuego3.java
```

```java
public static ArrayList<ArrayList<Soldier>> generateArmyB(ArrayList<ArrayList<Soldier>> a) {
    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 && a.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;
}
```

- 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, eso se comprobó cuando se decidió compilar el método showByCreation.
- Básicamente este método es igual que generateArmy, solo que este método 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, por lo que en la estructura condicional se añade esta línea a.get(row).get(column) == null

Listing 14: Compilando y probando las correcciones

```
javac VideoJuego3.java
java VideoJuego3
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]

DATOS DEL EJERCITO B
Soldier [name=Soldier3x5, lifePoints=3, row=4, column=6]
Soldier [name=Soldier5x8, lifePoints=1, row=6, column=9]
Soldier [name=Soldier8x1, lifePoints=1, row=9, column=2]
Soldier [name=Soldier9x6, lifePoints=5, row=10, column=7]
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 |___||___||___||___||___||_b_||___||___||___||___|
5 |___||___||___||___||___||___||___||___||___||___|
6 |___||___||___||___||___||___||___||___||_b_||___|
7 |___||___||___||___||___||___||_a_||___||___||___|
8 |___||___||___||___||___||___||___||___||___||___|
9 |___||_b_||___||___||___||___||___||___||___||___|
10|___||___||___||___||___||___||_b_||___||___||___|
```

- Para aclarar, en el main se corrigió la forma en la que se crea el ArrayList para el ejército B

Listing 15: Commit: Metodo que genera un arreglo bidimensional para el ejercito B

```
git add VideoJuego3.java
git commit -m "Metodo que genera un arreglo bidimensional para el ejercito B"
git push -u origin main
```

Listing 16: Se implementa el método que busca al soldado con mayor puntaje de vida

```
notepad++ VideoJuego3.java
```

```
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
```

- Se hace uso del método de ordenamiento Insertion.

```
135     public static Soldier longerLife(ArrayList<Soldier> s) {
136         int n = s.size();
137         for (int i = 1; i < n; i++) {
138             Soldier key = s.get(i);
139             int j = i - 1;
140             while (j >= 0 && s.get(j).getLifePoints() > key.getLifePoints()) {
141                 s.set(j + 1, s.get(j));
142                 j--;
143             }
144             s.set(j + 1, key);
145         }
146         return s.get(s.size() - 1);
147     }
```

- 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.

Listing 17: Compilando y probando

```
javac VideoJuego3.java
java VideoJuego
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=2, row=1, column=2]
Soldier [name=Soldier2x5, lifePoints=3, row=3, column=6]
Soldier [name=Soldier6x3, lifePoints=2, row=7, column=4]
Soldier [name=Soldier7x4, lifePoints=1, row=8, column=5]
Soldier [name=Soldier8x8, lifePoints=5, row=9, column=9]
Soldier [name=Soldier9x5, lifePoints=3, row=10, column=6]
Mayor vida en A: Soldier [name=Soldier8x8, lifePoints=5, row=9, column=9]

DATOS DEL EJRCITO B
Soldier [name=Soldier1x0, lifePoints=2, row=2, column=1]
Soldier [name=Soldier5x1, lifePoints=5, row=6, column=2]
Soldier [name=Soldier6x6, lifePoints=4, row=7, column=7]
Soldier [name=Soldier8x0, lifePoints=2, row=9, column=1]
Mayor vida en B: Soldier [name=Soldier5x1, lifePoints=5, row=6, column=2]
oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B    C    D    E    F    G    H    I    J
1 |___||_a_||___||___||___||___||___||___||___||___|
2 |_b_||___||___||___||___||___||___||___||___||___|
3 |___||___||___||___||___||_a_||___||___||___||___|
4 |___||___||___||___||___||___||___||___||___||___|
5 |___||___||___||___||___||___||___||___||___||___|
6 |___||_b_||___||___||___||___||___||___||___||___|
7 |___||___||___||_a_||___||___||_b_||___||___||___|
8 |___||___||___||___||_a_||___||___||___||___||___|
9 |_b_||___||___||___||___||___||___||___||_a_||___|
10|___||___||___||___||___||_a_||___||___||___||___|
```

Listing 18: Commit: Metodo longerLife culminado B

```
git add VideoJuego3.java
git commit -m "Metodo longerLife culminado"
git push -u origin main
```

Listing 19: Se implementa el método que retorna la vida total del ejército

```
notepad++ VideoJuego3.java
```

```java
    public static int totalLife(ArrayList<Soldier> sol) {
        int addition = 0;
        for (Soldier n : sol) {
            addition = addition + n.getLifePoints();
        }
        return addition;

    }
```

- El método devuelve los puntos de vida total del ejército. Se decide hacer el método con retorno para poder usar esos valores en otro método que nos permitirá determinar al ganador.

Listing 20: Commit: Metodo totalLife culminado B

```
git add VideoJuego3.java
git commit -m "Metodo totalLife culminado"
git push -u origin main
```

Listing 21: Se implementa el código que calculará el promedio de vida de cada ejército

```
notepad++ VideoJuego3.java
```

```java
public class VideoJuego3 {
    public static void main(String[] args) {

        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();
        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());
```

- Lo que está en azul, al costado de los números, es la implementación del código para calcular el promedio.

Listing 22: Compilando y probando

```
javac VideoJuego3.java
java VideoJuego
ooooooooooooooo FASE 1 DE LA CONTIENDA ooooooooooooooo
```

```
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=Soldier7x4, lifePoints=2, row=8, column=5]
Mayor vida en A: Soldier [name=Soldier1x4, lifePoints=4, row=2, column=5]
El total de vida del ejercito A es: 9
El promedio de vida del ejercito A es: 3.0

DATOS DEL EJRCITO B
Soldier [name=Soldier4x4, lifePoints=1, row=5, column=5]
Soldier [name=Soldier7x5, lifePoints=4, row=8, column=6]
Soldier [name=Soldier9x8, lifePoints=5, row=10, column=9]
Mayor vida en B: Soldier [name=Soldier9x8, lifePoints=5, row=10, column=9]
El total de vida del ejercito B es: 10
El promedio de vida del ejercito B es: 3.3333333333333335
ooooooooooooooo FASE 2 DE LA CONTIENDA ooooooooooooooo
Mostrando el tablero de juego
    A     B    C    D    E    F    G    H    I    J
1 |___||_a_||___||___||___||___||___||___||___||___|
2 |___||___||___||___||_a_||___||___||___||___||___|
3 |___||___||___||___||___||___||___||___||___||___|
4 |___||___||___||___||___||___||___||___||___||___|
5 |___||___||___||___||_b_||___||___||___||___||___|
6 |___||___||___||___||___||___||___||___||___||___|
7 |___||___||___||___||___||___||___||___||___||___|
8 |___||___||___||___||_a_||_b_||___||___||___||___|
9 |___||___||___||___||___||___||___||___||___||___|
10|___||___||___||___||___||___||___||___||_b_||___|
```

Listing 23: Commit:Codigo para calcular el promedio de vida para cada ejercito

```
git add VideoJuego3.java
git commit -m "Codigo para calcular el promedio de vida para cada ejercito"
git push -u origin main
```

Listing 24: Se implementa el método que ordena el ArrayList mostrando un ranking

```
notepad++ VideoJuego3.java
```

```
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
```

- Para este método se hará uso del método de ordenamiento Burbuja

```
176     public static void orderByPower(ArrayList<Soldier> sol) {
177         boolean swapped;
178         Soldier temp;
179         for (int i = 0; i < sol.size() - 1; i++) {
180             swapped = false;
181             for (int j = 0; j < sol.size() - 1 - i; j++) {
182                 temp = sol.get(j);
183                 sol.set(j, sol.get(j + 1));
184                 sol.set(j + 1, temp);
185                 swapped = true;
186             }
187             if (swapped == false) {
188                 break;
189             }
190         }
191         for (Soldier n : sol) {
192             System.out.println(n);
193         }
194     }
195
196 }
```

- El método ordena el ArrayList y se imprime el ranking, por lo que no retorna nada

Listing 25: Compilando y probando

```
javac VideoJuego3.java
java VideoJuego
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=Soldier0x7, lifePoints=5, row=1, column=8]
Soldier [name=Soldier1x1, lifePoints=5, row=2, column=2]
Soldier [name=Soldier4x9, lifePoints=1, row=5, column=10]
Soldier [name=Soldier5x7, lifePoints=2, row=6, column=8]
Soldier [name=Soldier8x9, lifePoints=3, row=9, column=10]
Mayor vida en A: Soldier [name=Soldier1x1, lifePoints=5, row=2, column=2]
El total de vida del ejercito A es: 16
El promedio de vida del ejercito A es: 3.2
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier1x1, lifePoints=5, row=2, column=2]
Soldier [name=Soldier0x7, lifePoints=5, row=1, column=8]
Soldier [name=Soldier8x9, lifePoints=3, row=9, column=10]
Soldier [name=Soldier5x7, lifePoints=2, row=6, column=8]
Soldier [name=Soldier4x9, lifePoints=1, row=5, column=10]

DATOS DEL EJRCITO B
Soldier [name=Soldier3x4, lifePoints=1, row=4, column=5]
Soldier [name=Soldier5x9, lifePoints=4, row=6, column=10]
Soldier [name=Soldier7x6, lifePoints=1, row=8, column=7]
Soldier [name=Soldier7x8, lifePoints=5, row=8, column=9]
Soldier [name=Soldier9x9, lifePoints=4, row=10, column=10]
Mayor vida en B: Soldier [name=Soldier7x8, lifePoints=5, row=8, column=9]
El total de vida del ejercito B es: 15
El promedio de vida del ejercito B es: 3.0
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier7x8, lifePoints=5, row=8, column=9]
Soldier [name=Soldier9x9, lifePoints=4, row=10, column=10]
Soldier [name=Soldier5x9, lifePoints=4, row=6, column=10]
Soldier [name=Soldier7x6, lifePoints=1, row=8, column=7]
Soldier [name=Soldier3x4, lifePoints=1, row=4, 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 |___||___||___||___||___||___||___||_a_||___||___|
2 |___||_a_||___||___||___||___||___||___||___||___|
3 |___||___||___||___||___||___||___||___||___||___|
4 |___||___||___||___||_b_||___||___||___||___||___|
5 |___||___||___||___||___||___||___||___||___||_a_|
6 |___||___||___||___||___||___||___||_a_||___||_b_|
7 |___||___||___||___||___||___||___||___||___||___|
8 |___||___||___||___||___||___||_b_||___||_b_||___|
9 |___||___||___||___||___||___||___||___||___||_a_|
10|___||___||___||___||___||___||___||___||___||_b_|
```

Listing 26: Commit:Se ha implementado el metodo orderByPower en el main

```
git add VideoJuego3.java
git commit -m "Se ha implementado el metodo orderByPower en el main"
git push -u origin main
```

Listing 27: Se implementa el método que determina al ganador

```
notepad++ VideoJuego3.java
```

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

- Para determinar a un ganador, se opta por enfrentarlos de acuerdo a su cantidad de puntos de vida.

```java
public class VideoJuego3 {
    public static void main(String[] args) {

        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("oooooooooooooooo  FASE 1 DE LA CONTIENDA  oooooooooooooooo");
        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();
        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();

        System.out.println("oooooooooooooooo  FASE 2 DE LA CONTIENDA  oooooooooooooooo");
        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);
    }
```

- En esta parte del código se tiene al Main el cual contiene la llamada de los métodos. Finalmente así quedaría el código.

Listing 28: Compilando y probando el codigo en su versión final

```
javac VideoJuego3.java
java VideoJuego
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=Soldier0x2, lifePoints=5, row=1, column=3]
Soldier [name=Soldier0x4, lifePoints=3, row=1, column=5]
Soldier [name=Soldier1x0, lifePoints=1, row=2, column=1]
Soldier [name=Soldier2x2, lifePoints=3, row=3, column=3]
Soldier [name=Soldier4x6, lifePoints=3, row=5, column=7]
Soldier [name=Soldier4x9, lifePoints=3, row=5, column=10]
Soldier [name=Soldier6x2, lifePoints=2, row=7, column=3]
Soldier [name=Soldier8x9, lifePoints=3, row=9, column=10]
Soldier [name=Soldier9x2, lifePoints=5, row=10, column=3]
Mayor vida en A: Soldier [name=Soldier9x2, lifePoints=5, row=10, column=3]
```

```
El total de vida del ejercito A es: 28
El promedio de vida del ejercito A es: 3.111111111111111
Mostrando soldados por ranking de poder de A
Soldier [name=Soldier9x2, lifePoints=5, row=10, column=3]
Soldier [name=Soldier0x2, lifePoints=5, row=1, column=3]
Soldier [name=Soldier8x9, lifePoints=3, row=9, column=10]
Soldier [name=Soldier4x9, lifePoints=3, row=5, column=10]
Soldier [name=Soldier4x6, lifePoints=3, row=5, column=7]
Soldier [name=Soldier2x2, lifePoints=3, row=3, column=3]
Soldier [name=Soldier0x4, lifePoints=3, row=1, column=5]
Soldier [name=Soldier6x2, lifePoints=2, row=7, column=3]
Soldier [name=Soldier1x0, lifePoints=1, row=2, column=1]

DATOS DEL EJRCITO B
Soldier [name=Soldier0x3, lifePoints=4, row=1, column=4]
Soldier [name=Soldier0x7, lifePoints=1, row=1, column=8]
Soldier [name=Soldier1x1, lifePoints=2, row=2, column=2]
Soldier [name=Soldier4x3, lifePoints=4, row=5, column=4]
Soldier [name=Soldier6x6, lifePoints=5, row=7, column=7]
Soldier [name=Soldier9x9, lifePoints=2, row=10, column=10]
Mayor vida en B: Soldier [name=Soldier6x6, lifePoints=5, row=7, column=7]
El total de vida del ejercito B es: 18
El promedio de vida del ejercito B es: 3.0
Mostrando soldados por ranking de poder de B
Soldier [name=Soldier6x6, lifePoints=5, row=7, column=7]
Soldier [name=Soldier4x3, lifePoints=4, row=5, column=4]
Soldier [name=Soldier0x3, lifePoints=4, row=1, column=4]
Soldier [name=Soldier9x9, lifePoints=2, row=10, column=10]
Soldier [name=Soldier1x1, lifePoints=2, row=2, column=2]
Soldier [name=Soldier0x7, lifePoints=1, row=1, column=8]

oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B    C    D    E    F    G    H    I    J
1 |___||___||_a_||_b_||_a_||___||___||_b_||___||___|
2 |_a_||_b_||___||___||___||___||___||___||___||___|
3 |___||___||_a_||___||___||___||___||___||___||___|
4 |___||___||___||___||___||___||___||___||___||___|
5 |___||___||___||_b_||___||___||_a_||___||___||_a_|
6 |___||___||___||___||___||___||___||___||___||___|
7 |___||___||_a_||___||___||___||_b_||___||___||___|
8 |___||___||___||___||___||___||___||___||___||___|
9 |___||___||___||___||___||___||___||___||___||_a_|
10|___||___||_a_||___||___||___||___||___||___||_b_|

++++++++++++++++++ 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 10 puntos de vida
```

Listing 29: Commit:Se ha aumentado codigo en el main y el metodo theWinner

```
git add VideoJuego3.java
git commit -m "Se ha aumentado codigo en el main y el metodo theWinner"
git push -u origin main
```

4.2. Estructura de laboratorio 06

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

```
lab06
|- -Soldier.java
|- -VideoJuego3.java
|
 ----latex
   |   programacion_lab06_rescobedoq_v1.0.pdf
   |   programacion_lab06_rescobedoq_v1.0.tex
   |
    ----img
         1.jpg
         10.jpg
         2.jpg
         3.jpg
         4.jpg
         5.jpg
         6.jpg
         7.jpg
         8.jpg
         9.jpg
         burbuja.jpg
         insertion.jpg
         logo_abet.png
         logo_episunsa.png
         logo_unsa.jpg
         main.jpg
```

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/