

Informe de Laboratorio 05

Tema: Arreglos Bidimensionales de Objetos

Nota

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

Laboratorio	Tema	Duración
05	Arreglos Bidimensionales de Objetos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 11 de Octubre 2023	Al 16 de octubre 2023

1. Tarea

- Inicializar el tablero con n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: Soldado0, Soldado1, 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. Además de los datos del Soldado con mayor vida, el promedio de puntos de vida de todos los soldados creados, el nivel de vida de todo el ejército, los datos de todos los soldados en el orden que fueron creados y un ranking de poder de todos los soldados creados, del que tiene más nivel de vida al que tiene menos (usar al menos 2 algoritmos de ordenamiento)
- 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.
- Arreglos Bidimensionales de objetos

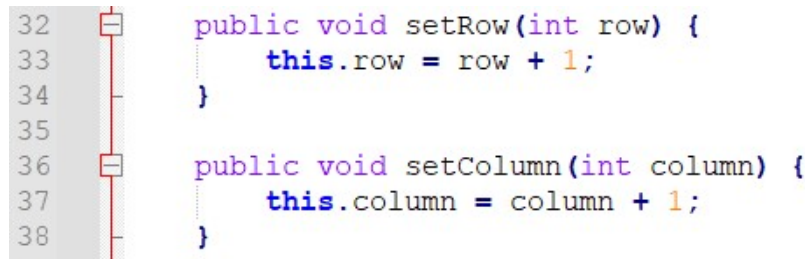
3. URL de Repositorio Github

- 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/lab05>

4. Actividades con el repositorio GitHub

Listing 1: Se inicializa el espacio de trabajo y se modifica la clase Soldier.java

```
mkdir lab05
cd lab03
Copy-Item "Soldier.java" -Destination "..\lab05"
Copy-Item "VideoJuego.java" -Destination "..\lab05"
cd ..
cd lab05
notepad++ Soldier.java
```



```
32 public void setRow(int row) {
33     this.row = row + 1;
34 }
35
36 public void setColumn(int column) {
37     this.column = column + 1;
38 }
```

- Se han borrado líneas de código del lab03 para poder realizar la actividad lab05
- En la clase Soldier.java se aumentan los atributos row y column junto con sus Getters y Setters. En los Setters de row y column se le suma uno ya que el usuario está acostumbrado a que se empiece a contar desde la posición uno y no desde la cero como lo hace el lenguaje de programación.

Listing 2: Commit: "Se ha adaptado VideoJuego.java del lab03 para el lab05, pero no se ha avanzado nada"

```
git add VideoJuego.java
git commit -m git commit -m "Se ha adaptado VideoJuego.java del lab03 para el lab05
pero no se ha avanzado nada"
git push -u origin main
git add Soldier.java
git add Soldier.java
git commit -m git commit -m "La clase Soldier.java esta terminada"
git push -u origin main
```

Listing 3: Se crea un método para generar el arreglo bidimensional de tipo Soldier

```
notepad++ VideoJuego.java
```

```

47 public static Soldier[][] generateArmy() {
48     Random random = new Random();
49     int amount = random.nextInt(10) + 1;
50     Soldier[][] myArmy = new Soldier[10][10];
51     int i = 0;
52     do {
53         int row = random.nextInt(10);
54         int column = random.nextInt(10);
55         if (myArmy[row][column] == null) {
56             String name = "Soldier" + (i + 1);
57             int lifePoints = random.nextInt(5) + 1;
58
59             myArmy[row][column] = new Soldier();
60
61             myArmy[row][column].setLifePoints(lifePoints);
62             myArmy[row][column].setName(name);
63             myArmy[row][column].setColumn(column);
64             myArmy[row][column].setRow(row);
65
66             i++;
67         }
68     } while (i < amount);
69     return myArmy;
70 }

```

- En el método generateArmy() se crea un arreglo bidimensional de 10 filas y 10 columnas con el fin de cubrir la misma cantidad de casillas del tablero según la práctica del laboratorio, por lo que habrán posiciones que quedarán vacías ya que la cantidad de elementos del arreglo será una cantidad 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

Listing 4: Commit: Se ha creado el metodo que permite generar un arreglo de soldados sin repetir posiciones

```

git VideoJuego.java
git commit -m git commit -m "Se ha creado el metodo que permite generar un arreglo de
soldados sin repetir posiciones"
git push -u origin main

```

Listing 5: Se crea un metodo que genera un arreglo unidimensional

notepad++ VideoJuego.java

```

24 public static Soldier[] arrayUnidimensional(Soldier[][] s) {
25     int contador = 0;
26     for (int i = 0; i < s.length; i++) {
27         for (int j = 0; j < s[i].length; j++) {
28             if (s[i][j] != null) {
29                 contador++;
30             }
31         }
32     }

```

- Este método se crea con el propósito de generar un arreglo unidimensional a partir del arreglo bidimensional ya creado, ya que nos facilitará el trabajo con los demás métodos que la práctica de

laboratorio requiere. De hecho, antes de crear este método, había otro que podía imprimir los elementos del arreglo bidimensional, pero fue reemplazado porque era necesario. Además, el método imprimir ya contenía líneas de código que permitían la creación del arreglo unidimensional, por lo que habría repetición de código

Listing 6: Commit: Se reemplaza el metodo de imprimir por el de generar arreglo unidimensional para poder trabajar los demas metodos

```
git add VideoJuego.java
git commit -m git commit -m " Se reemplaza el metodo de imprimir por el de generar
arreglo unidimensional para poder trabajar los demas metodos"
git push -u origin main
```

Listing 7: Método para imprimir el tablero con la posición de cada soldado

notepad++ VideoJuego.java

```
72 public static void miTablero(Soldier[][] a) {
73     String[][] tablero = new String[10][10];
74     for (int i = 0; i < tablero.length; i++) {
75         for (int j = 0; j < tablero[i].length; j++) {
76             tablero[i][j] = "|_|";
77         }
78     }
79
80     for (int i = 0; i < a.length; i++) {
81         for (int j = 0; j < a[i].length; j++) {
82             if (a[i][j] != null) {
83                 tablero[i][j] = "|_s_|";
84             }
85         }
86     }
87
88     System.out.print("    A    B    C    D    E    F    G    H    I    J \n");
89     for (int i = 0; i < tablero.length; i++) {
90         System.out.printf("%2d", (i + 1));
91         for (int j = 0; j < tablero[i].length; j++) {
92             System.out.print(tablero[i][j]);
93         }
94         System.out.println();
95     }
96 }
```

- Esta es la versión final del método. Las primeras versiones generaban un tablero bidimensional y en el main se hacía el reemplazo de los caracteres a "s", como también las entradas del tablero, sin embargo, para que haya un orden, se decide que en el metodo miTablero ya se pueda mostrar el tablero con los soldados ubicados en él y las entradas que están en el borde del tablero
- En el main se escribe la línea de código que hace llamado al método

Listing 8: Probando el tablero

```
javac VideoJuego.java
java VideoJuego
    A    B    C    D    E    F    G    H    I    J
1 | _ | _ | _ | _ | _ | s | _ | _ | _ | _ | _ | _ |
```

```

2 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ s _ | _ _ _ |
3 | _ s _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
4 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
5 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
6 | _ _ _ | _ s _ | _ _ _ | _ s _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
7 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
8 | _ s _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
9 | _ _ _ | _ s _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
10| _ s _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |

```

Listing 9: Commit: Se ha organizado la forma en la que se muestra el tablero, ahora está en un método

```

git add VideoJuego.java
git commit -m "Se ha organizado la forma en la que se muestra el tablero,
ahora esta en un metodo"
git push -u origin main

```

Listing 10: Método de ordenamiento por inserción y también la impresión de los datos de acuerdo a su orden de creación

notepad++ VideoJuego.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 |

```

- Este vendría a ser el pseudocódigo del método de ordenamiento por inserción

```

98 // Se hará uso del ordenamiento por inserción
99 public static void mayorVida(Soldier[] s) {
100     int n = s.length;
101     for (int i = 1; i < n; i++) {
102         Soldier key = s[i];
103         int j = i - 1;
104         while (j >= 0 && s[j].getLifePoints() > key.getLifePoints()) {
105             s[j + 1] = s[j];
106             j--;
107         }
108         s[j + 1] = key;
109     }
110 }

```

- Este vendría a ser el algoritmo de ordenamiento por inserción, pero adaptado a las necesidades del problema. Se implementa este algoritmo para que pueda ordenar el arreglo de menor a mayor ya que en una de las pautas del problema se nos exige usar dos algoritmos de ordenamiento, este vendría a ser el primero. En el main se escribe código el cual mostrará el último elemento del arreglo

```

112  public static void mostrarPorCreacion(Soldier[] sol) {
113      for (int i = 0; i < sol.length; i++) {
114          System.out.println(sol[i]);
115      }
116  }

```

- Como ya se mencionó, el método de mostrar los datos del arreglo Soldier ya estaba culminado, sin embargo por razones ya explicadas se decide cambiarlo e implementarlo después. En esta versión se opta por imprimir los datos de un arreglo unidimensional

```

8  public class VideoJuego {
9      public static void main(String[] args) {
10
11          Soldier[][] army = generateArmy();
12          Soldier[] armyU = arrayUnidimensional(army);
13          miTablero(army);
14          System.out.println("Mostrando soldados por orden de creacion");
15          mostrarPorCreacion(armyU);
16          mayorVida(armyU);
17          System.out.println("El soldado con mayor vida: " + armyU[armyU.length - 1]);

```

- En el main están las líneas de código que permitirán mostrar como funcionan estos dos métodos

Listing 11: Compilación de los métodos creados hasta el momento

```

javac VideoJuego.java
java VideoJuego
  A  B  C  D  E  F  G  H  I  J
1 |__|__|__|__|__|__|__|__|__|__|_s_|__|
2 |__|__|__|__|__|__|__|__|__|__|_s_|__|
3 |__|__|__|__|__|__|__|__|__|__|_s_|__|
4 |__|__|__|__|__|__|__|__|__|__|__|__|
5 |__|__|_s_|__|__|__|__|__|__|__|__|__|
6 |__|__|__|__|__|__|__|__|__|__|__|__|
7 |__|__|__|__|__|__|__|__|__|__|__|__|
8 |_s_|__|__|__|_s_|__|__|__|__|__|__|__|
9 |__|__|_s_|__|__|__|__|__|__|__|__|__|
10|__|__|__|__|__|__|__|__|__|__|__|__|_s_|
Mostrando soldados por orden de creacion
Soldier [name=Soldier7, lifePoints=5, row=1, column=9]
Soldier [name=Soldier5, lifePoints=2, row=2, column=9]
Soldier [name=Soldier3, lifePoints=5, row=3, column=7]
Soldier [name=Soldier2, lifePoints=4, row=5, column=2]
Soldier [name=Soldier8, lifePoints=3, row=8, column=1]
Soldier [name=Soldier6, lifePoints=5, row=8, column=3]
Soldier [name=Soldier4, lifePoints=1, row=9, column=2]
Soldier [name=Soldier1, lifePoints=4, row=10, column=10]
El soldado con mayor vida: Soldier [name=Soldier6, lifePoints=5, row=8, column=3]

```


Listing 12: Se crea el método que calcula el promedio y puntos de vida total del ejército

notepad++ VideoJuego.java

```
118 public static void totalLifeAndAverage(Soldier[] sol) {
119     int addition = 0;
120     for (int i = 0; i < sol.length; i++) {
121         addition = addition + sol[i].getLifePoints();
122     }
123     System.out.println("El promedio de vida del ejercito es: " + (double) addition / sol.length);
124     System.out.println("El total de vida del ejercito es: " + addition);
125 }
```

- Debido a que el promedio de vida del ejército se calcula en base a la suma de este, se decide que ambos valores se calculen en un mismo método y se impriman desde ya en el mismo método. De la misma manera que se hizo con los demás métodos, en el main se escribe una línea de código que llama a este método

Listing 13: Compilando y probando el método totalLifeAndAverage y demás

```
javac VideoJuego.java
java VideoJuego
  A  B  C  D  E  F  G  H  I  J
1 |__|__|__|__|__|__|__|__|_s_|__|__|__|
2 |__|__|__|__|_s_|__|__|__|__|__|__|__|
3 |__|_s_|__|__|__|__|__|__|__|__|__|__|
4 |_s_|__|__|__|__|__|__|__|__|__|__|__|
5 |__|__|__|__|__|__|__|__|__|__|__|__|
6 |__|__|__|__|__|__|__|__|__|__|__|__|
7 |__|__|__|__|__|__|__|__|__|__|__|__|
8 |__|__|__|__|__|__|__|__|__|__|__|__|
9 |__|__|__|__|__|_s_|__|__|_s_|__|__|__|
10|__|_s_|__|__|__|__|__|__|__|__|__|__|
Mostrando soldados por orden de creacion
Soldier [name=Soldier7, lifePoints=5, row=1, column=8]
Soldier [name=Soldier2, lifePoints=2, row=2, column=4]
Soldier [name=Soldier1, lifePoints=5, row=3, column=2]
Soldier [name=Soldier3, lifePoints=5, row=4, column=1]
Soldier [name=Soldier5, lifePoints=4, row=9, column=5]
Soldier [name=Soldier6, lifePoints=2, row=9, column=7]
Soldier [name=Soldier4, lifePoints=3, row=10, column=2]
El soldado con mayor vida: Soldier [name=Soldier3, lifePoints=5, row=4, column=1]
El promedio de vida del ejercito es: 3.7142857142857144
El total de vida del ejercito es: 26
```

Listing 14: Commit: "Se ha creado el metodo que calcula el promedio y total de los puntos de vida"

```
git add VideoJuego.java
git commit -m "Se ha creado el metodo que calcula el promedio y total de los puntos de vida"
git push -u origin main
```

Listing 15: Método burbuja que muestra el ranking de poder (vida)

notepad++ VideoJuego.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

```

- Este es el pseudocódigo del ordenamiento burbuja

```

129  public static void orderByPower(Soldier[] ar) {
130      boolean swapped;
131      Soldier temp;
132      for (int i = 0; i < ar.length - 1; i++) {
133          swapped = false;
134          for (int j = 0; j < ar.length - 1 - i; j++) {
135              temp = ar[j];
136              ar[j] = ar[j + 1];
137              ar[j + 1] = temp;
138              swapped = true;
139          }
140          if (swapped == false) {
141              break;
142          }
143      }
144      for (int i = 0; i < ar.length; i++) {
145          System.out.println(ar[i]);
146      }
147  }
148
149  }
150
151  }

```

- Este es el método burbuja ya implementado a las necesidades del código. En el método se decide incluir la impresión de los elementos ordenados ya que el método debe mostrar un ranking de acuerdo a la vida

Listing 16: Compilando y probando todos los métodos solicitados en la práctica

```

      A   B   C   D   E   F   G   H   I   J
1 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ s _ | _ _ _ | _ _ _ |
2 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
3 | _ _ _ | _ _ _ | _ _ _ | _ s _ | _ _ _ | _ _ _ | _ _ _ |
4 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
5 | _ _ _ | _ _ _ | _ _ _ | _ s _ | _ _ _ | _ _ _ | _ _ _ |
6 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
7 | _ s _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
8 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
9 | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
10| _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ s _ | _ _ _ | _ _ _ | _ s _ |
Mostrando soldados por orden de creacion
Soldier [name=Soldier3, lifePoints=3, row=1, column=6]
Soldier [name=Soldier6, lifePoints=3, row=3, column=4]
Soldier [name=Soldier5, lifePoints=1, row=5, column=4]
Soldier [name=Soldier4, lifePoints=1, row=7, column=1]
Soldier [name=Soldier1, lifePoints=3, row=10, column=5]
Soldier [name=Soldier2, lifePoints=4, row=10, column=10]
El soldado con mayor vida: Soldier [name=Soldier2, lifePoints=4, row=10, column=10]
El promedio de vida del ejercito es: 2.5
El total de vida del ejercito es: 15
Mostrando soldados por ranking de poder
Soldier [name=Soldier2, lifePoints=4, row=10, column=10]
Soldier [name=Soldier1, lifePoints=3, row=10, column=5]
Soldier [name=Soldier6, lifePoints=3, row=3, column=4]
Soldier [name=Soldier3, lifePoints=3, row=1, column=6]
Soldier [name=Soldier4, lifePoints=1, row=7, column=1]
Soldier [name=Soldier5, lifePoints=1, row=5, column=4]

```

Listing 17: Commit: Metodo burbuja que muestra el ranking de poder

```

git add VideoJuego.java
git commit -m git commit -m "Metodo burbuja que muestra el ranking de poder"
git push -u origin main

```

Listing 18: Estructura del main

```

notepad++ VideoJuego.java

```

```

8 public class VideoJuego {
9     public static void main(String[] args) {
10
11         Soldier[][] army = generateArmy();
12         Soldier[] armyU = arrayUnidimensional(army);
13         miTablero(army);
14         System.out.println("Mostrando soldados por orden de creacion");
15         mostrarPorCreacion(armyU);
16         mayorVida(armyU);
17         System.out.println("El soldado con mayor vida: " + armyU[armyU.length - 1]);
18         totalLifeAndAverage(armyU);
19         System.out.println("Mostrando soldados por ranking de poder");
20         orderByPower(armyU);
21     }
22 }

```

4.1. Estructura de laboratorio 05

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

```

lab05
| Soldier.java
| VideoJuego.java

+-----latex
| | programacion_lab05_rescobedoq_v1.0.pdf
| | programacion_lab05_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
|
+-----src

```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

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

5.2. Rúbrica para el contenido del Informe y demostración

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

Tabla 2: Niveles de desempeño

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

Tabla 3: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	3	
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	1	
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		17	

6. Referencias

- <https://www.geeksforgeeks.org/bubble-sort/>
- <https://www.geeksforgeeks.org/insertion-sort/>
- https://drive.google.com/file/d/18wvjXuguiRaIZ3Z0dElzC-LM9hrabhue/view?usp=drive_link