Informe de Laboratorio 05

Tema: Arreglos bidimensionales de objetos

Nota

Estudiante	Escuela	Asignatura
Ryan Fabian Valdivia Segovia rvaldiviase@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la programación 2 Semestre: II Código: 1701213

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

1.1. Videojuego

- Cree un Proyecto llamado Laboratorio5.
- Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo desarrollado en Laboratorio 3 y 4.
- Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el tablero debe ser un arreglo bidimensional de objetos.
- 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).

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Home Single Language 64 bits 22621.2283
- VIM 9.0.
- Visual Studio Code 64 bits 1.82.2
- OpenJDK 64-Bits 11.0.16.1
- Git 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/RyanValdivia/fp2-23b.git`
- URL para el laboratorio 05 en el Repositorio GitHub.
- `https://github.com/RyanValdivia/fp2-23b/tree/main/fase02/lab05`

4. Actividades

4.1. Actividad 1

- En primer lugar, realicé un commit conteniendo el código de la clase Soldado.java, requerido para la actividad principal.

Listing 1: Comentando el código de Soldado.java

```
$ git log lab05
commit a802a60b50724d01c56a86864d25c31caae71ce1
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date:  Sun Oct 15 21:13:11 2023 -0500
   Creando la clase 'Soldado.java' y trabajando el codigo principal, inicializando el
       arreglo de objetos bidimensional
```

- Conteniendo el siguiente código

Listing 2: Clase Soldado

```
6  public class Soldado {
7      private String nombre;
8      private int vida;
9      private int fila;
10     private int columna;
11
12     public void setNombre(String s) {
13         this.nombre = s;
14     }
15
```

```
16      public void setVida(int n) {
17          this.vida = n;
18      }
19
20      public void setFila(int n) {
21          this.fila = n;
22      }
23
24      public void setColumna(int n) {
25          this.columna = n;
26      }
27
28      public String getNombre() {
29          return nombre;
30      }
31
32      public int getVida() {
33          return vida;
34      }
35
36      public int getFila() {
37          return fila;
38      }
39
40      public int getColumna() {
41          return columna;
42      }
43  }
```

- Seguidamente, comencé elaborando los métodos requeridos para el funcionamiento del videojuego, por lo tanto, según mi lógica, lo primero era desplegar a los soldados en localizaciones aleatorias, sin que dos soldados estén en la misma casilla.
- Para esto, se me ocurrio crear dos arreglos de numeros enteros con numeros aleatorios, para luego tomar pares ordenados y asi formar coordenadas para los soldados sin que se repitan.

Listing 3: Números aleatorios

```
44    public static int[] numerosRandom(int q) {
45        int[] nums = new int[q];
46        for (int i = 0; i < nums.length; i++) {
47            nums[i] = nums.length;
48        }
49        for (int i = 0; i < q; i++) {
50            int n;
51            do {
52                n = (int) (Math.random() * 10);
53            } while (estaEnArreglo(nums, n, i));
54            nums[i] = n;
55        }
56        return nums;
57    }
```

Listing 4: Comprobar que no se repitan

```java
public static boolean estaEnArreglo(int[] arreglo, int num, int indice) {
    for (int i = 0; i < indice; i++) {
        if (arreglo[i] == num) {
            return true;
        }
    }
    return false;
}
```

- Una vez ya hecho el sistema para obtener las coordenadas, realicé un método para inicializar el arreglo bidimensional, dadas las coordenadas, así estableciendo todos sus respectivos atributos a todos los objetos.

Listing 5: Inicializando el ejército

```java
public static void inicializarArreglo(Soldado[][] army, int[] filas, int[] columnas) {
    for (int i = 0; i < filas.length; i++) {
        int v = (int) ((Math.random() * 5) + 1);
        army[filas[i]][columnas[i]] = new Soldado();
        army[filas[i]][columnas[i]].setNombre("Soldado" + i);
        army[filas[i]][columnas[i]].setVida(v);
        army[filas[i]][columnas[i]].setFila(filas[i]);
        army[filas[i]][columnas[i]].setColumna(columnas[i]);
    }
}
```

- Una vez inicializado, en el siguiente commit, creé el método para mostrar todo el tablero, con separadores para que se vea ésteticamente adecuado.

Listing 6: Mostrar el tablero

```java
public static void mostrarTablero(Soldado[][] army) {
    String vacio = "      ";
    System.out.println(crearTecho());
    for (int i = 0; i < army.length; i++) {
        System.out.println(separadorSup());
        for (int j = 0; j < army.length; j++) {
            if (j == army.length - 1) {
                if (army[i][j] == null) {
                    System.out.print("| " + vacio + " |");
                } else {
                    System.out.print("| " + army[i][j].getNombre() + " |");
                }
            } else {
                if (army[i][j] == null) {
                    System.out.print("| " + vacio + " ");
                } else {
                    System.out.print("| " + army[i][j].getNombre() + " ");
                }
            }
        }
        System.out.println();
        System.out.println(separadorInf());
```

```
98          }
99      }
```

- Para que esto se vea bonito de forma estética, creé varios métodos para poder armar el tablero.

Listing 7: Método para la fila superior del todo

```
100     public static String crearTecho() {
101         String franky = "";
102         for (int i = 0; i < 111; i++) {
103             franky += "_";
104         }
105         return franky;
106     }
```

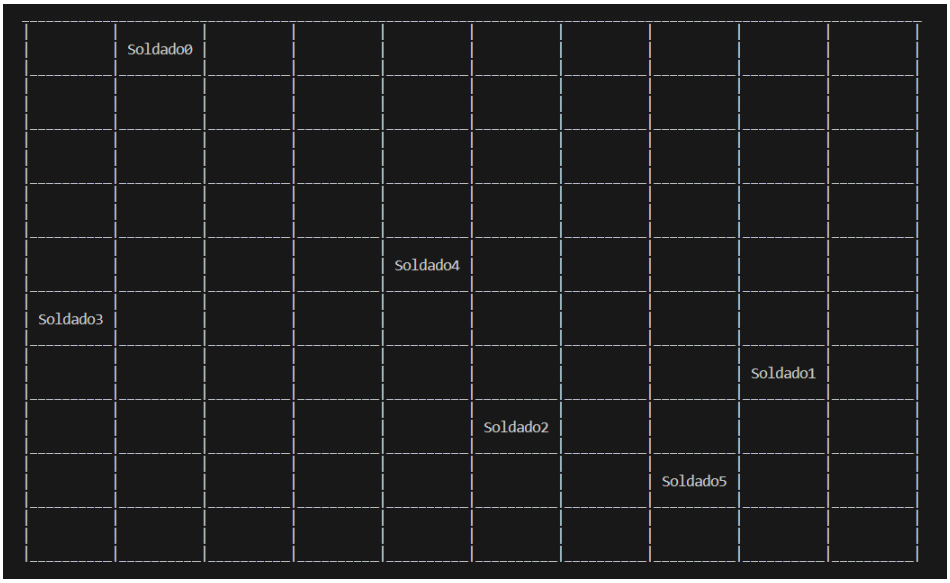
Listing 8: Método para separar las lineas(Parte superior)

```
107     public static String separadorSup() {
108         String franky = "";
109         for (int i = 0; i < 111; i++) {
110             if (i % 11 == 0) {
111                 System.out.print("|");
112             } else {
113                 System.out.print(" ");
114             }
115         }
116         return franky;
117     }
```

Listing 9: Método para separar las lineas(Parte inferior)

```
118     public static String separadorInf() {
119         String franky = "";
120         for (int i = 0; i < 111; i++) {
121             if (i % 11 == 0) {
122                 System.out.print("|");
123             } else {
124                 System.out.print("_");
125             }
126         }
127         return franky;
128     }
```

- Imprimiendo esto al momento de ejecutar el código.



- Luego de esto, en el siguiente commit, trabajé el método para determinar el soldado con mayor vida del arreglo.
- Lo hice simplemente recorriendo todos los soldados y obteniendo los indices del soldado con mayor vida.

Listing 10: Soldado con mayor nivel de vida

```
129 public static void soldadoMayorVida(Soldado[][] army, int[] filas, int[] columnas) {
130     int max = 0;
131     for (int i = 0; i < filas.length; i++) {
132         if (army[filas[i]][columnas[i]].getVida() >
                army[filas[max]][columnas[max]].getVida()) {
133             max = i;
134         }
135     }
136     System.out.println("El soldado con mayor vida es: ");
137     mostrarSoldado(army, filas[max], columnas[max]);
138 }
```

- Siguiendo con la ejecución de la anterior captura, nos imprimiría lo siguiente.

```
El soldado con mayor vida es:
Nombre: Soldado0
Vida: 4 HP
Posición: 1-B
```

- Además, por motivos de conveniencia, decidí crear un método para mostrar todos los datos de un soldado en la consola, siéndome útil para los siguientes métodos que requieran mostrar datos.

Listing 11: Mostrar un soldado

```
139  public static void mostrarSoldado(Soldado[][] army, int i, int j) {
140      String fila;
141      System.out.println("Nombre: " + army[i][j].getNombre());
142      System.out.println("Vida: " + army[i][j].getVida() + " HP");
143      switch (army[i][j].getColumna() + 1) {
144          case 1:
145              fila = "A";
146              break;
147          case 2:
148              fila = "B";
149              break;
150          case 3:
151              fila = "C";
152              break;
153          case 4:
154              fila = "D";
155              break;
156          case 5:
157              fila = "E";
158              break;
159          case 6:
160              fila = "F";
161              break;
162          case 7:
163              fila = "G";
164              break;
165          case 8:
166              fila = "H";
167              break;
168          case 9:
169              fila = "I";
170              break;
171          case 10:
172              fila = "J";
173              break;
174          default:
175              fila = "H";
176              break;
177      }
178      System.out.println("Posicion: " + (army[i][j].getFila() + 1) + "-" + fila);
179  }
```

- Ahora sigue un método para mostrar el nivel de vida de todo el ejército, así como el promedio.

Listing 12: Obtener la vida total del ejército y la vida promedio

```
180  public static double[] promedioYTotal(Soldado[][] army, int[] filas, int[] columnas) {
181      int total = 0;
182      for (int i = 0; i < filas.length; i++) {
183          total += army[filas[i]][columnas[i]].getVida();
184      }
185      double[] rpta = new double[] { total, (total * 1.0 / filas.length) };
186      return rpta;
187  }
```

- Esto devuelve un arreglo con dos valores, el valor total de vida del ejercito especificado y el valor promedio de vida por soldado, para poder imprimirlo después en el método main.

Listing 13: Método main hasta ahora

```
188  public static void main(String[] args) {
189      Scanner sc = new Scanner(System.in);
190      Soldado[][] ejercito = new Soldado[10][10];
191      int n = (int) ((Math.random() * 10) + 1);
192      int[] filas = numerosRandom(n);
193      int[] columnas = numerosRandom(n);
194      inicializarArreglo(ejercito, filas, columnas);
195      mostrarTablero(ejercito);
196      System.out.println();
197      soldadoMayorVida(ejercito, filas, columnas);
198      System.out.println("El nivel de vida de todo el ejercito es: " +
             promedioYTotal(ejercito, filas, columnas)[0]);
199      System.out.println("El nivel de vida promedio del ejercito es: " +
             promedioYTotal(ejercito, filas, columnas)[1]);
```

- Ahora sigue algo más sencillo, mostrar todos los soldados según el orden que fueron creados, para ello, solo debia recorrer el arreglo en base a las coordenadas que ya tenía.
- Por esto el método de 'mostrarSoldado' era necesario.

Listing 14: Mostrar ejército

```
200    public static void mostrarEjercito(Soldado[][] army, int[] filas, int[] columnas) {
201      System.out.println("Ejercito");
202      for (int i = 0; i < filas.length; i++) {
203          mostrarSoldado(army, filas[i], columnas[i]);
204          System.out.println();
205      }
206  }
```

- Imprimiendo lo siguiente al momento de ejecutar:

```
El nivel de vida de todo el ejercito es: 15.0
El nivel de vida promedio del ejercito es: 2.5

Ejercito
Nombre: Soldado0
Vida: 4 HP
Posición: 1-B

Nombre: Soldado1
Vida: 2 HP
Posición: 7-I

Nombre: Soldado2
Vida: 4 HP
Posición: 8-F

Nombre: Soldado3
Vida: 1 HP
Posición: 6-A

Nombre: Soldado4
Vida: 1 HP
Posición: 5-E

Nombre: Soldado5
Vida: 3 HP
Posición: 9-H
```

- Ahora seguía algo que ya hemos hecho en laboratorios anteriores, ordenar el ejército por orden de vida, por lo tanto, debíamos implementar algoritmos de ordenamiento necesarios.
- Lo primero que hice fue obtener un arreglo unidimensional para poder ordenarlo, en base al tablero que ya tenemos.

Listing 15: Obtener arreglo

```
207 public static Soldado[] crearArreglo(Soldado[][] army, int[] filas, int[] columnas) {
208     Soldado[] nuevo = new Soldado[filas.length];
209     for (int i = 0; i < nuevo.length; i++) {
210         nuevo[i] = army[filas[i]][columnas[i]];
211     }
212     return nuevo;
213 }
```

- Una vez obtenido el arreglo, solo queda programar los algoritmos de ordenamiento, para esto se pueden reutilizar códigos de clases anteriores.

Listing 16: Algoritmos de ordenamiento

```
215 public static void ordenamientoBurbuja(Soldado[] army) {
216     for (int i = 0; i < army.length; i++) {
217         for (int j = 0; j < army.length - 1; j++) {
218             if (army[j].getVida() > army[j + 1].getVida()) {
219                 intercambiar(army, j, j + 1);
220             }
221         }
222     }
223 }
224 public static void intercambiar(Soldado[] flota, int i, int j) {
225     Soldado temp;
226     temp = flota[i];
227     flota[i] = flota[j];
228     flota[j] = temp;
229 }
230
231 public static void ordenamientoInsercion(Soldado[] army) {
232     for (int i = 1; i < army.length; i++) {
233         Soldado valor = army[i];
234         int j = i;
235         for (j = i; 0 < j && army[j - 1].getVida() > valor.getVida(); j--) {
236             army[j] = army[j - 1];
237         }
238         army[j] = valor;
239     }
240 }
```

Listing 17: Metodo main (final)

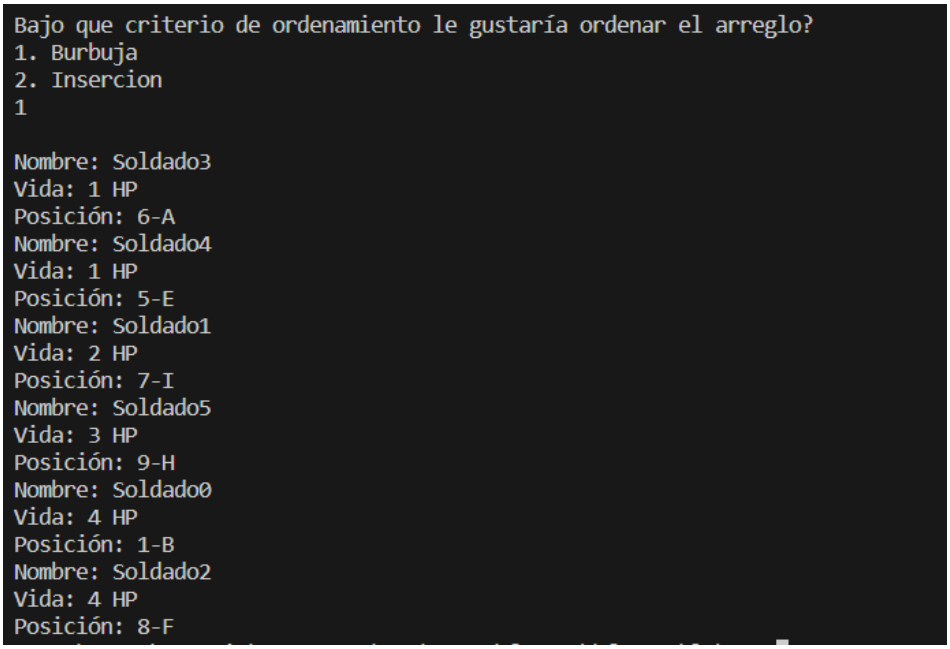
```
242  mostrarEjercito(ejercito, filas, columnas);
243     Soldado[] army = crearArreglo(ejercito, filas, columnas);
244     System.out.println("Bajo que criterio de ordenamiento le gustaria ordenar el
            arreglo?");
```

```
245         System.out.println("1. Burbuja");
246         System.out.println("2. Insercion");
247         switch (sc.nextInt()) {
248             case 1:
249                 ordenamientoBurbuja(army);
250                 break;
251             case 2:
252                 ordenamientoInsercion(army);
253                 break;
254             default:
255         }
256         System.out.println();
257         mostrar(army);
```

- Aqui solo llamé a todos los métodos desarrollados anteriormente, además de generar como un 'Menú' para que el usuario escoja cómo ordenar el ejército, con diferentes algoritmos de ordenamiento.
- En las siguientes capturas, podemos comprobar como funcionan los métodos de ordenamiento.

```
El nivel de vida de todo el ejercito es: 15.0
El nivel de vida promedio del ejercito es: 2.5

Ejercito
Nombre: Soldado0
Vida: 4 HP
Posición: 1-B

Nombre: Soldado1
Vida: 2 HP
Posición: 7-I

Nombre: Soldado2
Vida: 4 HP
Posición: 8-F

Nombre: Soldado3
Vida: 1 HP
Posición: 6-A

Nombre: Soldado4
Vida: 1 HP
Posición: 5-E

Nombre: Soldado5
Vida: 3 HP
Posición: 9-H
```

- Ordenando.



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	2	
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	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	4	
Total		20		18	

6. Referencias

- Fundamentos de la programación 2 - Tópicos de la programación Orientada a Objetos (Marco Aedo)