

Informe de Laboratorio 05

Tema: Arreglos Bidimensionales de Objetos

Nota			

Estudiante	Escuela	Asignatura
		Fundamentos de la
Christian Mestas Zegarra	Escuela Profesional de	Programación 2
cmestasz@unsa.edu.pe	Ingeniería de Sistemas	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 Octubre 2023	Al 16 Octubre 2023

1. Tarea

- Actividad 1: Cree un Proyecto llamado Laboratorio5.
- Actividad 2: Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo desarrollado en Laboratorio 3 y 4.
- Actividad 3: Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- Actividad 4: El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el tablero debe ser un arreglo bidimensional de objetos.
- Actividad 5: 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 (usar caracteres como | _ y otros). 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 Microsoft Windows 10 Pro 64 bits
- Visual Studio Code 1.82.2
- Java Development Kit 17.0.1
- Git 2.41.0.windows.1
- Windows PowerShell 5.1.19041.3031
- Cuenta en GitHub con el correo institucional.
- Arreglos bidimensionales de objetos.
- Ordenamientos burbuja y por selección.

3. URL de Repositorio Github

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

4. Actividades con el repositorio GitHub

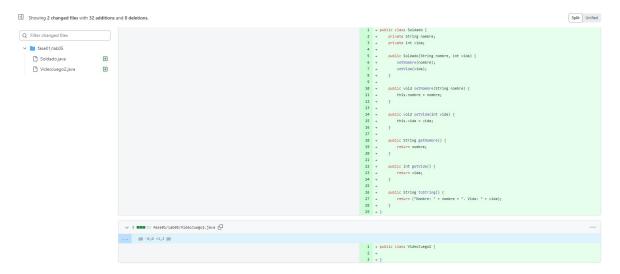
Creando plantillas

\$ mkdir lab05
\$ cd lab05
\$ code Soldado.java
\$ code VideoJuego2.java

Primer Commit / Plantillas







Primer Commit.

Actualizando VideoJuego2.java

\$ code VideoJuego2.java

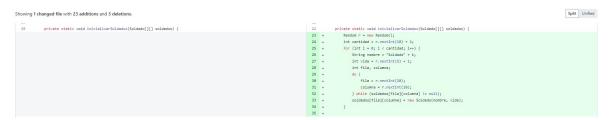
Segundo - Decimo Commit / VideoJuego2.java

```
$ git add .
$ git commit -m "Main y plantillas de metodos a utilizar"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodo inicializarSoldados()"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodo imprimirTablero(), incompleto"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodo imprimirTablero() y auxiliares"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Main, inicializarSoldados() y demas modificados"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodos soldadoMayorVida(), promedioPuntosVida(), sumaPuntosVida()"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodo ordenarSoldadosBurbuja()"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodo ordenarSoldadosSeleccion()"
$ code VideoJuego2.java
$ git add VideoJuego2.java
$ git commit -m "Metodo imprimirSoldados()"
$ git push
```

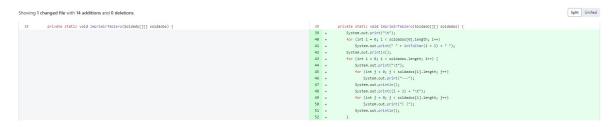




Segundo Commit.



Tercer Commit.



Cuarto Commit.

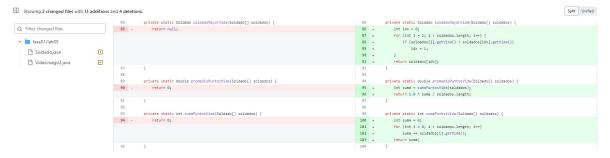


```
| Showing 1 charged file with 37 additions and 13 deletions. | Spite |
```

Quinto Commit.



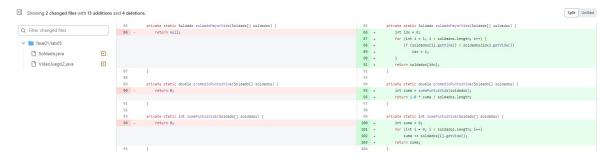
Sexto Commit.



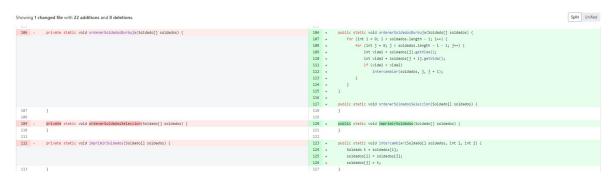
Séptimo Commit.



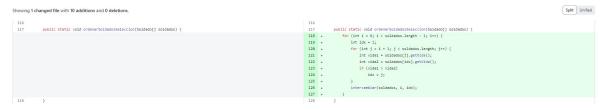




Octavo Commit.



Noveno Commit.



Decimo Commit.



5. Código desarrollado

Soldado.java

```
package fase01.lab05;
   public class Soldado {
       private String nombre;
       private int vida;
       public Soldado(String nombre, int vida) {
           setNombre(nombre);
           setVida(vida);
10
       public void setNombre(String nombre) {
12
           this.nombre = nombre;
14
       public void setVida(int vida) {
16
           this.vida = vida;
17
18
19
       public String getNombre() {
20
           return nombre;
21
       public int getVida() {
           return vida;
26
27
       public String toString() {
28
           return (nombre + " - " + vida);
29
       }
30
31
   }
```

- Clase que guarda nombre y vida del soldado.
- Posee tanto setters como getters para todos los atributos.
- Posee el metodo toString() para poder imprimir el objeto.



VideoJuego2.java

```
package fase01.lab05;
   import java.util.Random;
   public class VideoJuego2 {
       public static void main(String[] args) {
           Random r = new Random();
           int cantidad = r.nextInt(10) + 1;
           Soldado[][] tablero = new Soldado[10][10];
           Soldado[] soldados = new Soldado[cantidad];
           inicializarSoldados(tablero, soldados);
           imprimirTablero(tablero);
           System.out.printf("Soldado con mayor vida: %s %n", soldadoMayorVida(soldados));
13
           System.out.printf("Promedio de puntos de vida: %f %n", promedioPuntosVida(soldados));
           System.out.printf("Nivel de vida del ejercito: %d %n", sumaPuntosVida(soldados));
           Soldado[] soldados2 = new Soldado[soldados.length];
           Soldado[] soldados3 = new Soldado[soldados.length];
17
           System.arraycopy(soldados, 0, soldados2, 0, soldados.length);
18
           System.arraycopy(soldados, 0, soldados3, 0, soldados.length);
19
           System.out.println();
20
           ordenarSoldadosBurbuja(soldados2);
21
           imprimirSoldados(soldados2);
           System.out.println();
           ordenarSoldadosSeleccion(soldados3);
24
           imprimirSoldados(soldados3);
26
       public static void inicializarSoldados(Soldado[][] tablero, Soldado[] soldados) {
           Random r = new Random();
           for (int i = 0; i < soldados.length; i++) {</pre>
30
              String nombre = "Soldado" + i;
              int vida = r.nextInt(5) + 1;
32
              int fila, columna;
33
              do {
34
                  fila = r.nextInt(10);
                  columna = r.nextInt(10);
37
              } while (tablero[fila][columna] != null);
              Soldado soldado = new Soldado (nombre, vida);
              tablero[fila][columna] = soldado;
               soldados[i] = soldado;
           }
       }
       public static void imprimirTablero(Soldado[][] soldados) {
44
           System.out.print(generarEncabezado(soldados));
45
           String separacion = generarSeparacion(soldados);
46
           for (int i = 0; i < soldados.length; i++) {</pre>
47
               System.out.print(separacion);
               System.out.print(generarFila(soldados, i));
           System.out.print(separacion);
       }
       public static String generarEncabezado(Soldado[][] soldados) {
           String encabezado = "\t";
```





```
for (int i = 0; i < soldados.length; i++)</pre>
56
                encabezado += (" " + intToChar(i + 1) + " ");
            encabezado += " \n";
58
            return encabezado;
59
        }
60
61
        public static String generarSeparacion(Soldado[][] soldados) {
62
            String fila = "\t";
63
            for (int i = 0; i < soldados[0].length; i++)</pre>
64
                fila += "----";
65
            fila += "-\n";
66
            return fila;
        }
        public static String generarFila(Soldado[][] soldados, int f) {
            String fila = (f + 1) + "\t";
            for (int i = 0; i < soldados[f].length; i++) {</pre>
                fila += "| ";
73
                Soldado soldado = soldados[f][i];
74
                if (soldado != null)
75
                   fila += soldado.getNombre().charAt(soldado.getNombre().length() - 1);
76
                else
                   fila += " ";
                fila += " ";
            }
            fila += "|\n";
            return fila;
83
84
        public static Soldado soldadoMayorVida(Soldado[] soldados) {
85
            int idx = 0;
86
            for (int i = 1; i < soldados.length; i++) {</pre>
87
                if (soldados[i].getVida() > soldados[idx].getVida())
                    idx = i;
90
            return soldados[idx];
91
        }
92
93
        public static double promedioPuntosVida(Soldado[] soldados) {
            int suma = sumaPuntosVida(soldados);
            return 1.0 * suma / soldados.length;
97
98
        public static int sumaPuntosVida(Soldado[] soldados) {
99
            int suma = 0;
100
            for (int i = 0; i < soldados.length; i++)</pre>
                suma += soldados[i].getVida();
            return suma;
        }
104
        public static void ordenarSoldadosBurbuja(Soldado[] soldados) {
106
            for (int i = 0; i < soldados.length - 1; i++) {</pre>
                for (int j = 0; j < soldados.length - i - 1; <math>j++) {
                    int vida1 = soldados[j].getVida();
                    int vida2 = soldados[j + 1].getVida();
                    if (vida1 < vida2)</pre>
```



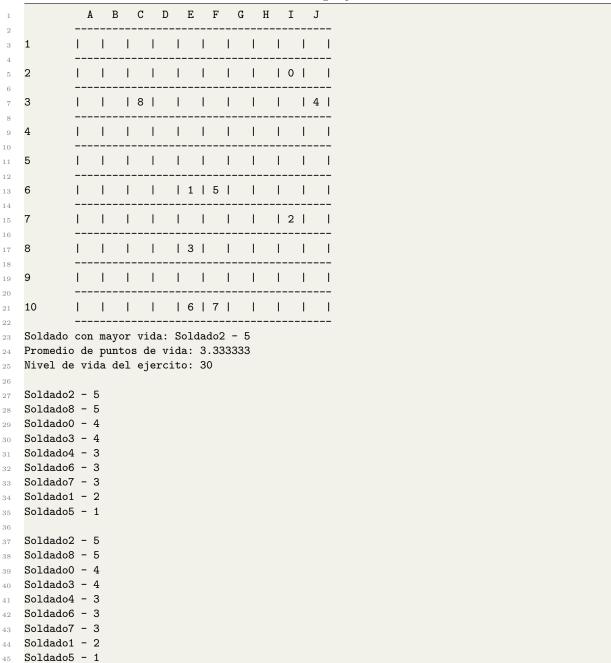
```
intercambiar(soldados, j, j + 1);
                }
113
            }
114
        public static void ordenarSoldadosSeleccion(Soldado[] soldados) {
117
            for (int i = 0; i < soldados.length - 1; i++) {</pre>
118
                int idx = i;
119
                for (int j = i + 1; j < soldados.length; j++) {</pre>
                    int vida1 = soldados[j].getVida();
                    int vida2 = soldados[idx].getVida();
                    if (vida1 > vida2)
                        idx = j;
                intercambiar(soldados, i, idx);
            }
        }
128
129
        public static void imprimirSoldados(Soldado[] soldados) {
130
            for (Soldado soldado : soldados)
                System.out.println(soldado);
        }
134
        public static void intercambiar(Soldado[] soldados, int i, int j) {
135
            Soldado t = soldados[i];
            soldados[i] = soldados[j];
            soldados[j] = t;
138
140
        public static char intToChar(int n) {
141
            return (char) (n + 'A' - 1);
142
143
    }
144
```

- Método inicializarSoldados() crea a los soldados, los ubica en el tablero y los guarda en un arreglo de soldados por separado.
- Método imprimirTablero() imprime el tablero con ayuda de los métodos auxiliares generarEncabezado(), generarSeparacion() y generarFila(), ubicando a los soldados por su numero.
- Método soldadoMayorVida() retorna el soldado con mayor vida.
- Método promedioPuntosVida() retorna el promedio de los puntos de vida de todos los soldados.
- Método sumaPuntosVida() retorna la suma de los puntos de vida de todos los soldados.
- Se crean dos copias del arreglo de soldados para demostrar los 2 ordenamientos.
- Método ordenarSoldadosBurbuja() ordena los soldados por vida de mayor a menor, usando ordenamiento burbuja.
- Método ordenarSoldadosSeleccion() ordena los soldados por vida de mayor a menor, usando ordenamiento selección.
- Método imprimirSoldados() imprime los soldados del ejercito.



6. Ejecución del código

VideoJuego2.java





7. Estructura de laboratorio 05

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

```
lab05/
|--- Soldado.java
|--- VideoJuego2.java
|--- ejec01.bash
|--- Informe.tex
|--- Informe.pdf
|--- img
   |--- logo_abet.png
   |--- logo_episunsa.png
   |--- logo_unsa.jpg
   |--- commit01.jpg
   |--- commit02.jpg
   |--- commit03.jpg
   |--- commit04.jpg
   |--- commit05.jpg
   |--- commit06.jpg
   |--- commit07.jpg
   |--- commit08.jpg
   |--- commit09.jpg
   |--- commit10.jpg
```



8. Rúbricas

8.1. Entregable Informe

Tipo de Informe

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

8.2. 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 aprobatoria, siempre y cuando cumpla con todos los items.
- El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

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



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	1.5	
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	1.5	
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	

9. Referencias

■ Aedo, M. y Castro, E. (2021). FUNDAMENTOS DE PROGRAMACIÓN 2 - Tópicos de Programación Orientada a Objetos. Editorial UNSA.