

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la programación II				
TÍTULO DE LA PRÁCTICA:	Arreglos Bidimensionales de Objetos				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2024	NRO. SEMESTRE:	02
FECHA DE PRESENTACIÓN	18/10/2024	HORA DE PRESENTACIÓN	22/10/00		
INTEGRANTE (s) <i>José León Enrique Hatches Curo</i>				NOTA (0-20)	<i>Nota colocada por el docente</i>
DOCENTE(s): <i>Ing. Lino Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p><i>El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.</i></p> <p><i>El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado.</i></p> <p>Repositorio en GitHub:</p> <p>https://github.com/LeonHatches/Laboratorio05</p>

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

VIDEOJUEGO 2:

Inicializar el tablero con “n” soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: “Soldado0”, “Soldado1”, etc. Un valor de nivel de vida autogenerado aleatoriamente [1 ... 5], la fila y columna también autogenerados aleatoriamente (verificar que 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, mostrar los datos del Soldado con mayor nivel de vida, el promedio de nivel 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).

- *Primero, este es el main con todos los métodos.*

```
public static void main (String [] args)
{
    Soldado [] tableroUni;
    Soldado [][] tablero = new Soldado [10][10];

    // Arreglos bidimensionales
    tableroUni = crear (tablero);
    mostrarTabla (tablero);
    mostrarMayorVida (tablero);



    double vida = mostrarPromedioVida (tablero);

    // MOSTRAR VIDA DE TODOS LOS SOLDADOS
    System.out.println ("\n| VIDA TOTAL DEL EJERCITO |");
    System.out.println ("La vida total es: " + vida );

    // Arreglos estandar
    System.out.println("\n| SOLDADOS - ORDEN DE CREACION |");
    mostrar (tableroUni);

    System.out.println("\n| SOLDADOS - RANKING DE VIDA MAYOR A MENOR |");
    mostrarRankingMayor (tableroUni);

    System.out.println("\n| SOLDADOS - RANKING DE VIDA MENOR A MAYOR |");
    mostrarRankingMenor (tableroUni);
}
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

- En este método se muestra la creación y verificación del objeto Soldado, además la inicialización de un arreglo estándar para los ordenamientos.

```
public static Soldado [] crear (Soldado [][] tablero)
{
    int aleatorio = posicionRandom()+1, fila, columna;
    Soldado [] tableroUni = new Soldado[aleatorio];

    for (int i = 0 ; i < aleatorio ; i++)
    {
        // VERIFICA POSICION LIBRE
        do
        {
            fila = posicionRandom();
            columna = posicionRandom();
        }
        while ( tablero[fila][columna] != null );

        // CREA E INICIALIZA DATOS DEL SOLDADO
        tablero[fila][columna] = new Soldado();
        inicializar (tablero, fila, columna, i);

        // ARREGLO ESTANDAR PARA TRABAJAR CON ORDENAMIENTO
        tableroUni[i] = new Soldado ();
        tableroUni[i] = tablero[fila][columna];
    }
    return tableroUni;
}
```

- Luego, este método inicializa los objetos con un nombre, vida, fila y columna según la tabla del enunciado.

```
public static void inicializar (Soldado [][] tablero, int f, int c, int cont)
{
    String columnaLetras [] ={"A","B","C","D","E","F","G","H","I","J"};

    tablero[f][c].setNombre ("Soldado"+cont);
    tablero[f][c].setVida ( vida() );
    tablero[f][c].setFila (f+1);
    tablero[f][c].setColumna (columnaLetras[c]);
}
```

- Luego, este método muestra la tabla con los objetos (solo nombre).

```
public static void mostrarTabla (Soldado [][] tablero)
{
    String letras [] = {"A","B","C","D","E","F","G","H","I","J"};
    System.out.print ("      ");

    // MUESTRA LAS COLUMNAS
    for (int i = 0 ; i < letras.length ; i++)
    {
        System.out.print (letras[i]+"      ");
    }
    System.out.println(" ");

    for (int i = 0 ; i < tablero.length ; i++)
    {
        // MUESTRA LAS FILAS
        System.out.print (i+1);
        for (int j = 0 ; j < tablero[i].length ; j++)
        {
            // SI NO EXISTE SOLDADO, MUESTRA UN ESPACIO EN BLANCO
            if (tablero[i][j] == null)
                System.out.print ("      |");
            else
                System.out.print (" "+tablero[i][j].getNombre()+" |");
        }
        System.out.println("\n-----"
            + "-----");
    }
}
```

- Después, estos muestran una posición y vida aleatoria.

```
public static int posicionRandom ()
{
    return (int) (Math.random() * 10);
}

public static int vida ()
{
    return (int) (Math.random() * 5 + 1);
}
```

- *Luego, este método muestra el soldado con mayor vida, con una verificación de los valores iniciales a tomar antes de todo.*

```
public static void mostrarMayorVida (Soldado[][]tablero)
{
    int fila = 0, columna = 0;

    // LOS MAYORES VALORES INICIALES
    for (int i = 0 ; i < tablero.length ; i++) {
        for (int j = 0 ; j < tablero[i].length ; j++)
        {
            if ( !(tablero[i][j] == null) )
            { fila = i; columna = j; break; }
        }
    }

    // BUSQUEDA DEL SOLDADO CON MAYOR VIDA
    for (int i = 0 ; i < tablero.length ; i++)
    {
        for (int j = 0 ; j < tablero[i].length ; j++){
            if ( !(tablero[i][j] == null) ) {
                if ( tablero[fila][columna].getVida() < tablero[i][j].getVida() )
                {
                    fila = i;
                    columna = j;
                }
            }
        }
    }

    // MOSTRAR
    System.out.println ("\\n\\t| SOLDADO CON MAYOR VIDA |\\n");
    System.out.println (tablero[fila][columna]+"\\n");
}
```

- Después, este método se encarga de mostrar el promedio de vida de todos los soldados y devolver la vida total del ejército.

```
public static double mostrarPromedioVida (Soldado[][] tablero)
{
    double suma = 0, cont = 0;

    // PROMEDIO DE VIDA
    for (int i = 0 ; i < tablero.length ; i++)
    {
        for (int j = 0 ; j < tablero[i].length ; j++){
            if ( !(tablero[i][j] == null) )
            {
                suma += tablero[i][j].getVida();
                cont++;
            }
        }
    }

    // MOSTRAR
    System.out.println ("\n\t| PROMEDIO DE VIDA |\n");
    System.out.println ("El promedio de vida es: " + (suma/cont) );

    return suma;
}
```

- En este método se hace un ordenamiento de inserción al arreglo estándar para mostrar un ranking de mayor a menor.

```
// Desde ahora, metodos con arreglo estandar para trabajar el ordenamiento o metodos que lo requieran.
public static void mostrarRankingMayor (Soldado [] tablero)
{
    Soldado intercambio;

    // Algoritmo de ordenamiento Insercion
    for (int i = 1 ; i < tablero.length ; i++)
    {
        for (int r = i ; r > 0 && tablero[r-1].getVida() < tablero[r].getVida() ; r--)
        {
            intercambio = tablero [r-1];
            tablero [r-1] = tablero [r];
            tablero [r] = intercambio;
        }
    }

    mostrar (tablero);
}
```

- *Luego, este es otro método de ordenamiento, pero con un algoritmo burbuja.*

```
public static void mostrarRankingMenor (Soldado [] tablero)
{
    Soldado intercambio;

    // Algoritmo de ordenamiento Burbuja
    for (int i = tablero.length - 1 ; i > 0 ; i--)
        for (int n = 0 ; n < i ; n++)
        {
            if ( tablero[n].getVida() > tablero[n+1].getVida() )
            {
                intercambio = tablero[n];
                tablero[n] = tablero[n+1];
                tablero[n+1] = intercambio;
            }
        }

    mostrar(tablero);
}
```

- *Por último, este método muestra los soldados del tablero en lista.*

```
public static void mostrar (Soldado [] tablero)
{
    for (int i = 0 ; i < tablero.length ; i++)
        System.out.println("\t| SOLDADO N-"+(i+1)+" | \n"+tablero[i]+" \n");
}
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

II. PRUEBAS

EJECUCIÓN:

GIT:

- Aquí hice mi primer commit que tiene "Soldado.java" y "VideoJuego2.java", en donde tiene los métodos de vida y posicionamiento aleatorio.

```

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio (main)
$ git add .

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Soldado.java
        new file:   VideoJuego2.java

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio (main)
$ git commit -m "Primer commit, clases incompletas."
[main (root-commit) 3d638e0] Primer commit, clases incompletas.
 2 files changed, 86 insertions(+)
 create mode 100644 Soldado.java
 create mode 100644 VideoJuego2.java

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio (main)
$ git remote add origin https://github.com/LeonHatches/Laboratorio05.git

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 874 bytes | 874.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/LeonHatches/Laboratorio05.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

```

- Este es mi 2do commit, en el cual tiene métodos como para mostrar la tabla con los datos inicializados.

```

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio/Laboratorio05 (main)
$ git commit -m "Segundo commit, hasta mostrar tablero con datos ya inicializados."
[main 8807889] segundo commit, hasta mostrar tablero con datos ya inicializados.
 2 files changed, 100 insertions(+), 69 deletions(-)

Usuario24B@DESKTOP-TBA9GB6 MINGW64 ~/desktop/repositorio/Laboratorio05 (main)
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.20 KiB | 1.20 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/LeonHatches/Laboratorio05.git
 3d638e0..8807889  main -> main
branch 'main' set up to track 'origin/main'.

```


- dc6e93e
- ☐
-

[illegible]

- Se muestra soldado con mayor vida, con promedio de vida y la vida total.

```
| SOLDADO CON MAYOR VIDA |
```

```
NOMBRE: Soldado2
FILA: 5
COLUMNA: G
VIDA: 3
```

```
| PROMEDIO DE VIDA |
```

```
El promedio de vida es: 1.5
```

```
| VIDA TOTAL DEL EJERCITO |
```

```
La vida total es: 6.0
```

- Se muestra el arreglo en orden de creación.

```
| SOLDADOS - ORDEN DE CREACION |
```

```
| SOLDADO N-1 |
```

```
NOMBRE: Soldado0
FILA: 1
COLUMNA: E
VIDA: 1
```

```
| SOLDADO N-2 |
```

```
NOMBRE: Soldado1
FILA: 3
COLUMNA: C
VIDA: 1
```

```
| SOLDADO N-3 |
```

```
NOMBRE: Soldado2
FILA: 5
COLUMNA: G
VIDA: 3
```

```
| SOLDADO N-4 |
```

```
NOMBRE: Soldado3
FILA: 10
COLUMNA: I
VIDA: 1
```

- *Se muestra el ranking de mayor a menor según su vida.*

```
| SOLDADOS - RANKING DE VIDA MAYOR A MENOR |
| SOLDADO N-1 |
NOMBRE: Soldado2
FILA: 5
COLUMNA: G
VIDA: 3

| SOLDADO N-2 |
NOMBRE: Soldado0
FILA: 1
COLUMNA: E
VIDA: 1

| SOLDADO N-3 |
NOMBRE: Soldado1
FILA: 3
COLUMNA: C
VIDA: 1

| SOLDADO N-4 |
NOMBRE: Soldado3
FILA: 10
COLUMNA: I
VIDA: 1
```

- *Y por último, se muestra el ranking de menor a mayor según su vida.*

```
| SOLDADOS - RANKING DE VIDA MENOR A MAYOR |
| SOLDADO N-1 |
NOMBRE: Soldado0
FILA: 1
COLUMNA: E
VIDA: 1

| SOLDADO N-2 |
NOMBRE: Soldado1
FILA: 3
COLUMNA: C
VIDA: 1

| SOLDADO N-3 |
NOMBRE: Soldado3
FILA: 10
COLUMNA: I
VIDA: 1

| SOLDADO N-4 |
NOMBRE: Soldado2
FILA: 5
COLUMNA: G
VIDA: 3
```

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 12

III. CUESTIONARIO:

Colocar la evidencia de las respuestas realizadas al cuestionario enunciado en la guía práctica de laboratorio.

Preguntas:

¿Con qué valores comprobaste que tu práctica estuviera correcta?

- *En este ejercicio, comprobé mediante datos o valores inicializados de tipo String e int.*

¿Qué resultado esperabas obtener para cada valor de entrada?

- *Esperaba los resultados de mostrar un soldado con mayor vida, promedio de vida, suma total de vida del tablero y mostrar los soldados según el algoritmo de ordenamiento.*

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

- *Obtuve los valores que deseaba para satisfacer el enunciado.*

CONCLUSIONES

En conclusión, para que el código del laboratorio pueda tener un buen funcionamiento, el uso de arreglos bidimensionales será necesario especialmente en el tablero y el uso de objetos para implementar los datos de los soldados. Además, la importancia del uso de métodos que devuelvan los arreglos ordenados de diferentes formas, así mostremos gran optimización, accesibilidad y orden al momento de programar. Así también, la importancia de los métodos al instante de trabajar y tener un código que tenga una facilidad de observación al momento de las correcciones del código o en este caso, la visualización de algoritmos de ordenamiento.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 13</p>

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

- *La secuencia que utilicé fue, primero, analizar el código base ya proporcionado o que había realizado y tratar de darle parte de la solución en mi mente. Segundo, me dediqué a armar la estructura de los algoritmos faltantes, los distintos métodos. Tercero, analizar errores y formas de optimizar el código para que se visualice de una manera entendible. Cuarto y último, compilar o correr el código para visualizar su funcionamiento o en dado caso no funcione, volver al paso 3 para corregir.*

REFERENCIAS Y BIBLIOGRAFÍA

Ninguna referencia externa.

RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió 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 1: 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

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	1	
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	0	
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		15	

Tabla 2: Rúbrica para contenido del Informe y demostración