INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Fundamentos de Programación II*				
TÍTULO DE LA PRÁCTICA:	*Arreglos Bidimensionales de Objetos*				
NÚMERO DE PRÁCTICA:	*5*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*II*
FECHA DE PRESENTACIÓN	*18/10/2024*	**HORA DE PRESENTACIÓN**	*5/20/00*		
INTEGRANTE (s) *Usiel Suriel Quispe Puma*				**NOTA (0-20)**	*Nota colocada por el docente*
DOCENTE(s): *Mg.Lino José Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

LINK DE MI REPOSITORIO : https://github.com/usiel33/Laboratorios_FP2

Ejercicio 1

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).

Código:

- **Clase VideoJuego (Prinicipal):**

```
package Laboratorio_05;
//Laboratorio 05- Ejercicio 1
//Autor:Usiel Suriel Quispe Puma
import java.util.*;

public class VideoJuego {

    public static void main(String[] args) {
        int cantSoldados;
        String nombre;
        int fil, col, nivelVida;

        // creamos un arreglo bidimensional de 10x10
        Soldado soldados[][] = new Soldado[10][10];

        Random rd = new Random();
        cantSoldados = rd.nextInt(10) + 1;
        ArrayList<Soldado> listaSoldados = new ArrayList<>();

        // Ingresamos todos los datos de los soldados
        for (int i = 0; i < cantSoldados; i++) {
            nombre = "Soldado" + i;
            nivelVida = rd.nextInt(5) + 1;

            // Asignamos el soldado en una posición aleatoria vacía
            while (true) {
                fil = rd.nextInt(10);
                col = rd.nextInt(10);

                if (soldados[fil][col] == null) {
                    Soldado nuevoSoldado = new Soldado(nombre, nivelVida, fil, col);
                    soldados[fil][col] = nuevoSoldado;
                    listaSoldados.add(nuevoSoldado);
                    break;
```

```
                }
            }
        }

        // Mostrar el tablero de los soldados
        mostrarTablero(soldados);

        // Mostrar el soldado con la mayor vida
        mostrarSoldadoMayorVida(soldados);

        // Mostrar promedio de vida de todos los soldados
        mostrarPromedioVida(soldados);

        // Mostrar el nivel de vida total del ejército
        mostrarNivelVidaEjercito(soldados);

        // Mostrar los soldados en el orden que fueron creados
        mostrarSoldados(listaSoldados);

        // Mostrar ranking de poder de los soldados
        rankearSoldadosPorVida(listaSoldados);
    }

    //Método para mostrar el tablero con todos los soldados creados
    public static void mostrarTablero(Soldado[][] soldados) {
        System.out.println("Tablero de soldados:\n");
        System.out.println("    A  B  C  D  E  F  G  H  I  J");
        for (int i = 0; i < soldados.length; i++) {
            System.out.print((i + 1) + " "); // Ajuste para que las filas inicien correctamente desde 1
            for (int j = 0; j < soldados[0].length; j++) {
                if (soldados[i][j] != null) {
                    System.out.print("|S| ");
                } else {
                    System.out.print("|_| ");
```

```
                }
            }
            System.out.println();
        }
    }

    // Método que muestra el soldado con la mayor vida
    public static void mostrarSoldadoMayorVida(Soldado[][] soldados) {
        Soldado soldadoMayor = null;
        int mayorNivelVida = 0;
        for (int i = 0; i < soldados.length; i++) {
            for (int j = 0; j < soldados[0].length; j++) {
                if (soldados[i][j] != null && soldados[i][j].getNivelVida() > mayorNivelVida) {
                    soldadoMayor = soldados[i][j];
                    mayorNivelVida = soldados[i][j].getNivelVida();
                }
            }
        }
        // Verificamos si encontramos algún soldado en la tabla
        if (soldadoMayor != null) {
            System.out.println("\nEl soldado con mayor nivel de vida es:");
            soldadoMayor.mostrarDatos();
        } else {
            System.out.println("\nNo hay soldados en el tablero.");
        }
    }

    // Método que muestra el promedio de nivel de vida de todos los soldados
    public static void mostrarPromedioVida(Soldado[][] soldados) {
        int sumaVida = 0;
        int contadorSoldados = 0;
        float promedio;

        for (int i = 0; i < soldados.length; i++) {
```

```
            for (int j = 0; j < soldados[0].length; j++) {
                if (soldados[i][j] != null) {
                    sumaVida += soldados[i][j].getNivelVida();
                    contadorSoldados++;
                }
            }
        }

        // Verificamos si hay soldados
        if (contadorSoldados > 0) {
            promedio = (float) sumaVida / contadorSoldados;
            System.out.println("\nEl promedio de vida de los soldados  : " + promedio);
        } else {
            System.out.println("\nNo hay soldados en el tablero ");
        }
    }

    // Método para mostrar el nivel de vida total de todo el ejército
    public static void mostrarNivelVidaEjercito(Soldado[][] soldados) {
        int sumaVida = 0;

        for (int i = 0; i < soldados.length; i++) {
            for (int j = 0; j < soldados[0].length; j++) {
                if (soldados[i][j] != null) {
                    sumaVida += soldados[i][j].getNivelVida();
                }
            }
        }

        System.out.println("\nNivel de vida de todo el ejercito: " + sumaVida);
    }

    // Método para mostrar los soldados en el orden en que fueron creados
    public static void mostrarSoldados(ArrayList<Soldado> soldados) {
        System.out.println("\nSoldados en orden de creacion :");
        for (Soldado soldado : soldados) {
            soldado.mostrarDatos();
            System.out.println();
        }
    }

    // Método para rankear los soldados por nivel de vida
    public static void rankearSoldadosPorVida(ArrayList<Soldado> soldados) {
        for (int i = 0; i < soldados.size() - 1; i++) {
            for (int j = 0; j < soldados.size() - i - 1; j++) {
                if (soldados.get(j).getNivelVida() < soldados.get(j + 1).getNivelVida()) {
                    Soldado temp = soldados.get(j);
                    soldados.set(j, soldados.get(j + 1));
                    soldados.set(j + 1, temp);
                }
            }
        }

        // Mostrar ranking
        System.out.println("Ranking de poder de los soldados ( mayor al menor nivel de vida):\n");
        for (Soldado soldado : soldados) {
            soldado.mostrarDatos();
            System.out.println("");
        }
    }
}
```

- **Clase Soldado:**

```
package Laboratorio_05;
//Laboratorio 05- Ejercicio 1- Clase Soldado
//Autor:Usiel Suriel Quispe Puma
public class Soldado {
    // Atributos privados
    private String nombre;
    private int nivelVida;
    private int fila;
    private int columna;

    public Soldado(String nombre, int nivelVida, int fila, int columna) {
        this.nombre = nombre;
        this.nivelVida = nivelVida;
        this.fila = fila;
        this.columna = columna;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getNivelVida() {
        return nivelVida;
    }

    public void setNivelVida(int nivelVida) {
        this.nivelVida = nivelVida;
    }

    public int getFila() {
        return fila;
    }

    public int getColumna() {
        return columna;
    }

    // Metodo para mostrar datos de cada soldado
    public void mostrarDatos() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Nivel de Vida: " + nivelVida);
        System.out.println("Posicion: Fila " + (fila+1) + ", Columna "
                + (char)(columna + 'A'));
    }
}
```

Pruebas:

```
Tablero de soldados:

    A  B  C  D  E  F  G  H  I  J
1 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
2 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
3 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
4 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
5 |_| |_| |_| |_| |S| |_| |_| |_| |_| |_|
6 |_| |_| |_| |_| |_| |_| |_| |S| |_| |_|
7 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
8 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
9 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
10 |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|

El soldado con mayor nivel de vida es:
Nombre: Soldado0
Nivel de Vida: 1
Posicion: Fila 5, Columna E

El promedio de vida de los soldados  : 1.0

Nivel de vida de todo el ejercito: 2

Soldados en orden de creacion :
Nombre: Soldado0
Nivel de Vida: 1
Posicion: Fila 5, Columna E

Nombre: Soldado1
Nivel de Vida: 1
Posicion: Fila 6, Columna H

Ranking de poder de los soldados ( mayor al menor nivel de vida):

Nombre: Soldado0
Nivel de Vida: 1
Posicion: Fila 5, Columna E

Nombre: Soldado1
Nivel de Vida: 1
Posicion: Fila 6, Columna H
```

- **Commit:**

```
Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git add src/Laboratorio_05

Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git commit -m "Se sube el laboratorio 5 a la carpeta src"
[master 2482ba9] Se sube el laboratorio 5 a la carpeta src
 2 files changed, 212 insertions(+)
 create mode 100644 src/Laboratorio_05/Soldado.java
 create mode 100644 src/Laboratorio_05/VideoJuego.java

Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.16 KiB | 315.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/usiel33/Laboratorios_FP2.git
   fe80e3f..2482ba9  master -> master

Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git log
commit 2482ba985b19659d47f946f5befa68c176288775 (HEAD -> master, origin/master, origin/HEAD)
Author: usiel33 <uquispep@unsa.edu.pe>
Date:   Fri Oct 18 02:26:43 2024 -0500

    Se sube el laboratorio 5 a la carpeta src
```

III. Rubrica

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		3	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4		2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2		2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2		2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2		2	
6. Fechas	Las fechas de modificación del código fuente	2		4	

	están dentro de los plazos de fecha de entrega establecidos.				
7. Ortografía	El documento no muestra errores ortográficos.	2		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		2	
TOTAL		20		18	

CONCLUSIONES

En este laboratorio vimos el uso de la poo, el arreglo de objeto y ordenamientos , los cuales son importante en el desarrollo de programas ya que mejora la legibilidad y permite separar el código en métodos para reutilizar códigos , además implementar algoritmos de ordenamientos , que ya están definidas, hace que estos procesos sean más eficientes.

METODOLOGÍA DE TRABAJO

primero revise el código ya escrito para poder entender el funcionamiento y los requerimientos que necesita, luego complete con código cada método haciendo que cumpla con el requerimiento.
finalmente hice pruebas para verificar el correcto funcionamiento del código .

REFERENCIAS Y BIBLIOGRAFÍA

Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE