


	<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: 1

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de Programación 2				
TÍTULO DE LA PRÁCTICA:	Combinando Arreglos Estándar y ArrayList				
NÚMERO DE PRÁCTICA:	7	AÑO LECTIVO:	2024-B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	15/11/2024	HORA DE PRESENTACIÓN	18/20/00		
INTEGRANTE (s)				NOTA (0-20)	
Riveros Vilca Alberth Edwar					
DOCENTE(s):					
Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <ol style="list-style-type: none"> 1. Cree un Proyecto llamado Laboratorio7 2. Usted deberá crear las dos clases Soldado.java y VideoJuego4.java. Puede reutilizar lo desarrollado en Laboratorios anteriores. 3. Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero). 4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada.

	<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>

5. Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, 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 y sus puntos de vida (usar caracteres como | _ y otros y distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa Iterativo.

CLASE SOLDADO:

```
public class Soldado {
    private String nombre;
    private int fila;
    private int columna;
    private int nivelVida;

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

    public void setNombre(String n) {
        nombre = n;
    }

    public void setFila(int f) {
        fila = f;
    }

    public void setColumna(int c) {
        columna = c;
    }

    public void setNivelVida(int p) {
        nivelVida = p;
    }

    public String getNombre() {
        return nombre;
    }

    public int getFila() {
        return fila;
    }

    public int getColumna() {
        return columna;
    }

    public int getNivelVida() {
        return nivelVida;
    }

    @Override
    public String toString() {
        return "[Nombre: " + this.getNombre() + "\tFila: " + (this.getFila() + 1) + "\tColumna: " + (this.getColumna() + 1)
            + "\tNivel de Vida: " + this.getNivelVida() + "]\n";
    }
}
```

codesnap.dev

CLASE VIDEOJUEGO4:

```
import java.util.*;
public class VideoJuego4 {
    public static void main(String[] args) {
        Random rand = new Random();
        boolean[][] casillasOcupadas = new boolean[10][10];
        Soldado[][] ejercito1 = new Soldado[10][10];
        Soldado[][] ejercito2 = new Soldado[10][10];
        ArrayList<String> nombresEjercito1 = new ArrayList<>();
        ArrayList<String> nombresEjercito2 = new ArrayList<>();
        int numSoldados1 = rand.nextInt(10) + 1;
        int numSoldados2 = rand.nextInt(10) + 1;

        crearEjercito(rand, numSoldados1, ejercito1, casillasOcupadas, nombresEjercito1, 1);
        crearEjercito(rand, numSoldados2, ejercito2, casillasOcupadas, nombresEjercito2, 2);

        showBoard(casillasOcupadas, ejercito1, ejercito2);

        System.out.println("Ejército 1:");
        mostrarDatosEjercito(ejercito1, nombresEjercito1, numSoldados1);
        System.out.println("Ejército 2:");
        mostrarDatosEjercito(ejercito2, nombresEjercito2, numSoldados2);
        determinarGanador(ejercito1, ejercito2);
    }
    /*Metodo que determina al ganador de acuerdo al total de vida de los ejércitos*/
    public static void determinarGanador(Soldado[][] ejercito1, Soldado[][] ejercito2) {
        int totalVidaEjercito1 = calcularTotalVida(ejercito1);
        int totalVidaEjercito2 = calcularTotalVida(ejercito2);

        System.out.printf("Total Vida Ejército 1: %d\n", totalVidaEjercito1);
        System.out.printf("Total Vida Ejército 2: %d\n", totalVidaEjercito2);

        if (totalVidaEjercito1 > totalVidaEjercito2) {
            System.out.println("El Ejército 1 gana la batalla.");
        } else if (totalVidaEjercito2 > totalVidaEjercito1) {
            System.out.println("El Ejército 2 gana la batalla.");
        } else {
            System.out.println("La batalla termina en empate.");
        }
    }
    /*Metodo que muestra todos los datos de un ejércitos almacenados en un arreglo bidimensional*/
    public static void mostrarDatosEjercito(Soldado[][] ejercito, ArrayList<String> nombresEjercito, int numSoldados) {
        String mayorVida = findMaxLifeSoldier(ejercito);
        System.out.println("Soldado de Mayor Vida: " + mayorVida);

        double promedioVida = calcularPromedioVida(ejercito, numSoldados);
        System.out.println("Promedio de nivel de vida: " + promedioVida);

        System.out.println("Soldados en el orden de creación:");
        armyCreationOrder(nombresEjercito);

        ArrayList<Soldado> soldadosLista = toList(ejercito);
        bubbleSortLife(soldadosLista);
        System.out.println("Ranking de poder (Bubble Sort):");
        showArmyInfo(soldadosLista);

        soldadosLista = toList(ejercito);
        insertionSortLife(soldadosLista);
        System.out.println("Ranking de poder (Insertion Sort):");
        showArmyInfo(soldadosLista);
    }
}
```

```
/*Metodo que crea un ejercito en un arreglo bidimensional*/
public static void crearEjercito(Random rand, int numSoldados, Soldado[][] ejercito, boolean[][] casillasOcupadas,
                                ArrayList<String> nombresEjercito, int ejercitoNum) {

    int count = 0;
    while (count < numSoldados) {
        int randColumn = rand.nextInt(10);
        int randRow = rand.nextInt(10);
        if (!casillasOcupadas[randRow][randColumn]) {
            casillasOcupadas[randRow][randColumn] = true;
            String nombreSoldado = "Soldado" + count + "X" + ejercitoNum;
            nombresEjercito.add(nombreSoldado);
            ejercito[randRow][randColumn] = new Soldado(nombreSoldado, randRow, randColumn,
                                                         rand.nextInt(5) + 1);
            count++;
        }
    }
}

/*Metodo que muestra el tablero con los soldados y sus vidas correspondientes*/
public static void showBoard(boolean[][] casillasOcupadas, Soldado[][] ejercito1, Soldado[][] ejercito2) {
    System.out.print("\t");
    for (char i = 'A'; i < 'K'; i++) {
        System.out.print(" " + i + " ");
    }
    System.out.println();
    System.out.print(" ");
    for (int l = 0; l < 10; l++) {
        System.out.print("-----");
    }
    System.out.println();
    for (int j = 0; j < 10; j++) {
        System.out.print((j + 1) + (j < 9 ? " " : " "));
        for (int k = 0; k < 10; k++) {
            System.out.print("|");
            if (casillasOcupadas[j][k]) {
                if (ejercito1[j][k] != null) {
                    System.out.printf(" :%-2d ", ejercito1[j][k].getNivelVida());
                } else if (ejercito2[j][k] != null) {
                    System.out.printf(" :%-2d ", ejercito2[j][k].getNivelVida());
                }
            } else {
                System.out.print(" ");
            }
        }
        System.out.println("|");
        System.out.print(" ");
        for (int l = 0; l < 10; l++) {
            System.out.print("-----");
        }
        System.out.println();
    }
}

/*Metodo que muestra el orden de creacion de un ejercito */
public static void armyCreationOrder(ArrayList<String> nombresEjercito) {
    for (String nombre : nombresEjercito) {
        System.out.println(nombre);
    }
    System.out.println();
}
}
```

```
/*Metodo que devuelve un arraylist de un ejercito almacenado en un arreglo bidimensional*/
public static ArrayList<Soldado> toList(Soldado[][] soldados) {
    ArrayList<Soldado> lista = new ArrayList<>();
    for (Soldado[] fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null) {
                lista.add(soldado);
            }
        }
    }
    return lista;
}

/*Metodo que retorna los atributos de los soldados de un ejercito almacenado en arraylist*/
public static void showArmyInfo(ArrayList<Soldado> ejercito) {
    for (Soldado soldado : ejercito) {
        System.out.print(soldado);
    }
}

/*Metodo que retorna el total de vida de un ejercito*/
public static int calcularTotalVida(Soldado[][] soldados) {
    int totalVida = 0;
    for (Soldado[] fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null) {
                totalVida += soldado.getNivelVida();
            }
        }
    }
    return totalVida;
}

/*Metodo que retorna el promedio de vida de un ejercito*/
public static double calcularPromedioVida(Soldado[][] soldados, int numSoldados) {
    int totalVida = calcularTotalVida(soldados);
    return (double) totalVida / numSoldados;
}

/*Metodo que encuentra el soldado con la mayor cantidad de vida de un ejercito*/
public static String findMaxLifeSoldier(Soldado[][] soldados) {
    Soldado max = null;
    for (Soldado[] fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null && (max == null || soldado.getNivelVida() > max.getNivelVida())) {
                max = soldado;
            }
        }
    }
    return max != null ? max.toString() : "No hay soldados.";
}

/*Metodo que realiza bubble sort con arraylist de soldados*/
public static void insertionSortLife(ArrayList<Soldado> ejercito) {
    for (int i = 1; i < ejercito.size(); i++) {
        Soldado key = ejercito.get(i);
        int j = i - 1;
        while (j >= 0 && ejercito.get(j).getNivelVida() > key.getNivelVida()) {
            ejercito.set(j + 1, ejercito.get(j));
            j--;
        }
        ejercito.set(j + 1, key);
    }
}

/*Metodo que realiza bubble sort con arraylist de soldados*/
public static void bubbleSortLife(ArrayList<Soldado> ejercito) {
    for (int i = 0; i < ejercito.size() - 1; i++) {
        for (int j = 0; j < ejercito.size() - i - 1; j++) {
            if (ejercito.get(j).getNivelVida() < ejercito.get(j + 1).getNivelVida()) {
                Soldado temp = ejercito.get(j);
                ejercito.set(j, ejercito.get(j + 1));
                ejercito.set(j + 1, temp);
            }
        }
    }
}
```

EJECUCIÓN:

	A	B	C	D	E	F	G	H	I	J
1	*:1			+:3		*:4				
2				+:1						
3					*:4		*:2		+:5	
4					+:3		*:3			
5					*:1		*:5	+:1		
6										
7		+:3			+:5			+:1		
8		*:1			+:5					
9			+:5					*:2		
10					*:4					

Ejército 1:

Soldado de Mayor Vida: [Nombre: Soldado5X1 Fila: 3 Columna: 9 Nivel de Vida: 5]

Promedio de nivel de vida: 3.2

Soldados en el orden de creación:

Soldado0X1
Soldado1X1
Soldado2X1
Soldado3X1
Soldado4X1
Soldado5X1
Soldado6X1
Soldado7X1
Soldado8X1
Soldado9X1

Ranking de poder (Bubble Sort):

[Nombre: Soldado5X1 Fila: 3 Columna: 9 Nivel de Vida: 5]
[Nombre: Soldado8X1 Fila: 7 Columna: 5 Nivel de Vida: 5]
[Nombre: Soldado6X1 Fila: 8 Columna: 5 Nivel de Vida: 5]
[Nombre: Soldado2X1 Fila: 9 Columna: 3 Nivel de Vida: 5]
[Nombre: Soldado3X1 Fila: 1 Columna: 4 Nivel de Vida: 3]
[Nombre: Soldado9X1 Fila: 4 Columna: 5 Nivel de Vida: 3]
[Nombre: Soldado4X1 Fila: 7 Columna: 2 Nivel de Vida: 3]
[Nombre: Soldado0X1 Fila: 2 Columna: 4 Nivel de Vida: 1]
[Nombre: Soldado1X1 Fila: 5 Columna: 8 Nivel de Vida: 1]
[Nombre: Soldado7X1 Fila: 7 Columna: 8 Nivel de Vida: 1]

Ranking de poder (Insertion Sort):

[Nombre: Soldado0X1 Fila: 2 Columna: 4 Nivel de Vida: 1]
[Nombre: Soldado1X1 Fila: 5 Columna: 8 Nivel de Vida: 1]
[Nombre: Soldado7X1 Fila: 7 Columna: 8 Nivel de Vida: 1]
[Nombre: Soldado3X1 Fila: 1 Columna: 4 Nivel de Vida: 3]
[Nombre: Soldado9X1 Fila: 4 Columna: 5 Nivel de Vida: 3]
[Nombre: Soldado4X1 Fila: 7 Columna: 2 Nivel de Vida: 3]
[Nombre: Soldado5X1 Fila: 3 Columna: 9 Nivel de Vida: 5]
[Nombre: Soldado8X1 Fila: 7 Columna: 5 Nivel de Vida: 5]
[Nombre: Soldado6X1 Fila: 8 Columna: 5 Nivel de Vida: 5]
[Nombre: Soldado2X1 Fila: 9 Columna: 3 Nivel de Vida: 5]

Ejército 2:

Soldado de Mayor Vida: [Nombre: Soldado2X2 Fila: 5 Columna: 7 Nivel de Vida: 5]

Promedio de nivel de vida: 2.7

Soldados en el orden de creación:

Soldado0X2

Soldado1X2

Soldado2X2

Soldado3X2

Soldado4X2

Soldado5X2

Soldado6X2

Soldado7X2

Soldado8X2

Soldado9X2

Ranking de poder (Bubble Sort):

[Nombre: Soldado2X2 Fila: 5 Columna: 7 Nivel de Vida: 5]

[Nombre: Soldado5X2 Fila: 1 Columna: 6 Nivel de Vida: 4]

[Nombre: Soldado0X2 Fila: 3 Columna: 5 Nivel de Vida: 4]

[Nombre: Soldado3X2 Fila: 10 Columna: 5 Nivel de Vida: 4]

[Nombre: Soldado6X2 Fila: 4 Columna: 7 Nivel de Vida: 3]

[Nombre: Soldado4X2 Fila: 3 Columna: 7 Nivel de Vida: 2]

[Nombre: Soldado1X2 Fila: 9 Columna: 8 Nivel de Vida: 2]

[Nombre: Soldado8X2 Fila: 1 Columna: 1 Nivel de Vida: 1]

[Nombre: Soldado7X2 Fila: 5 Columna: 5 Nivel de Vida: 1]

[Nombre: Soldado9X2 Fila: 8 Columna: 2 Nivel de Vida: 1]

Ranking de poder (Insertion Sort):

[Nombre: Soldado8X2 Fila: 1 Columna: 1 Nivel de Vida: 1]

[Nombre: Soldado7X2 Fila: 5 Columna: 5 Nivel de Vida: 1]

[Nombre: Soldado9X2 Fila: 8 Columna: 2 Nivel de Vida: 1]

[Nombre: Soldado4X2 Fila: 3 Columna: 7 Nivel de Vida: 2]

[Nombre: Soldado1X2 Fila: 9 Columna: 8 Nivel de Vida: 2]

[Nombre: Soldado6X2 Fila: 4 Columna: 7 Nivel de Vida: 3]

[Nombre: Soldado5X2 Fila: 1 Columna: 6 Nivel de Vida: 4]

[Nombre: Soldado0X2 Fila: 3 Columna: 5 Nivel de Vida: 4]



[Nombre: Soldado3X2 Fila: 10 Columna: 5 Nivel de Vida: 4]

[Nombre: Soldado2X2 Fila: 5 Columna: 7 Nivel de Vida: 5]

Total Vida Ejército 1: 32

Total Vida Ejército 2: 27

El Ejército 1 gana la batalla.

	<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: 9

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

Verifique que los datos se generarán correctamente.

¿Qué resultado esperabas obtener para cada valor de entrada?

Que los soldados tengan los atributos correctos como el nombre, número de fila y columna, y vida dentro de los valores permitidos.



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

Obtuve comportamientos correctos aunque tuve que ajustar el método de mostrar tablero por los nuevos requisitos.

III. CUESTIONARIO:

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

CAPTURAS DE LOS COMMIT:



	<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: 10</p>

```

rivxd@pop-os:~/Imágenes/U/Laboratorios FP2-Lino
~/Imágenes/U/Laboratorios FP2-Lino on main |15 72| git branch -M main ✓ (at 18:09:44)
~/Imágenes/U/Laboratorios FP2-Lino on main |15 72| git add . ✓ (at 18:10:05)
~/Imágenes/U/Laboratorios FP2-Lino on main +18 | git status ✓ (at 18:10:09)
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios a ser confirmados:
(usa "git restore --staged <archivo>..." para sacar del área de stage)
borrados: Labo06/.gitattributes
borrados: Labo06/.gitignore
borrados: Labo06/.vscode/settings.json
borrados: Labo06/app/bin/main/labo06/Soldado.class
borrados: Labo06/app/bin/main/labo06/VideoJuego2.class
borrados: Labo06/app/build.gradle
borrados: Labo06/app/src/main/java/labo06/Soldado.java
borrados: Labo06/app/src/main/java/labo06/VideoJuego2.java
borrados: Labo06/app/src/test/java/labo06/AppTest.java
borrados: Labo06/gradle/libs.versions.toml
borrados: Labo06/gradle/wrapper/gradle-wrapper.jar
borrados: Labo06/gradle/wrapper/gradle-wrapper.properties
borrados: Labo06/gradlew
borrados: Labo06/gradlew.bat
borrados: Labo06/settings.gradle
nuevos archivos: RIVEROS_VILCA_LABORATORIO_04/.idea/workspace.xml
nuevos archivos: RIVEROS_VILCA_LABORATORIO_07/Soldado.java
nuevos archivos: RIVEROS_VILCA_LABORATORIO_07/VideoJuego4.java

```

	<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: 11</p>

```

rivxd@pop-os:~/Imágenes/U/Laboratorios FP2-Lino
~/Im/U/Laboratorios FP2-Lino on main +18 git commit -m "lab07-finalizado"
[main 543a5d2] lab07-finalizado
18 files changed, 327 insertions(+), 662 deletions(-)
delete mode 100644 Labo06/.gitattributes
delete mode 100644 Labo06/.gitignore
delete mode 100644 Labo06/.vscode/settings.json
delete mode 100644 Labo06/app/bin/main/lab06/Soldado.class
delete mode 100644 Labo06/app/bin/main/lab06/VideoJuego2.class
delete mode 100644 Labo06/app/build.gradle
delete mode 100644 Labo06/app/src/main/java/lab06/Soldado.java
delete mode 100644 Labo06/app/src/main/java/lab06/VideoJuego2.java
delete mode 100644 Labo06/app/src/test/java/lab06/AppTest.java
delete mode 100644 Labo06/gradle/libs.versions.toml
delete mode 100644 Labo06/gradle/wrapper/gradle-wrapper.jar
delete mode 100644 Labo06/gradle/wrapper/gradle-wrapper.properties
delete mode 100755 Labo06/gradlew
delete mode 100644 Labo06/gradlew.bat
delete mode 100644 Labo06/settings.gradle
create mode 100644 RIVEROS_VILCA_LABORATORIO_04/.idea/workspace.xml
create mode 100644 RIVEROS_VILCA_LABORATORIO_07/Soldado.java
create mode 100644 RIVEROS_VILCA_LABORATORIO_07/VideoJuego4.java

~/Im/U/Laboratorios FP2-Lino on main ↗ git push -u origin main ✓ (at 18:10:32)
Username for 'https://github.com': rivX241
Password for 'https://rivX241@github.com':
Enumerando objetos: 10, listo.
Contando objetos: 100% (10/10), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (7/7), listo.
Escribiendo objetos: 100% (8/8), 3.93 KiB | 3.93 MiB/s, listo.
Total 8 (delta 2), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS.git
4e38829..543a5d2 main -> main
Rama 'main' configurada para hacer seguimiento a la rama remota 'main' de 'origin'.

~/Imágenes/U/Laboratorios FP2-Lino on main

```



Cambie la rama a main, añadí el informe al área de stage y hice un commit lab05-finalizado y realice el git

push -u origin main el origin ya estaba previamente configurado para todos los laboratorios.

LINK: https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS

CONCLUSIONES

Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.

	<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

Trabajar con arreglos bidimensionales y ArrayList unidimensionales en Java me hizo darme cuenta de lo prácticos que son para manejar datos de manera organizada. Es como tener un tablero donde puedo acceder a cada elemento con precisión usando índices. Sin embargo, también me enfrenté a ciertos desafíos, ya que recorrer y manipular estas estructuras requiere más cuidado. En el ejercicio que estoy desarrollando, los datos no están organizados de forma contigua, lo que complica la implementación de algunos algoritmos, pero también me reta a mejorar mis habilidades para manejar datos más complejos.

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.



Al abordar este problema, primero analicé los requisitos para comprender cómo manejar los arreglos bidimensionales y los niveles de vida de los soldados con ArrayList .

Luego estructuré el código, asegurándome de representar correctamente los valores como `+:nivelDeVida` y `*:nivelDeVida` en la salida. Durante las pruebas, ajusté la representación visual de la tabla para que fuera más clara y ordenada.

Finalmente, corregí errores y optimicé el diseño, aprendiendo cómo combinar eficientemente arreglos bidimensionales y ArrayList para resolver problemas complejos.

REFERENCIAS Y BIBLIOGRAFÍA

E. G. Castro Gutiérrez y M. W. Aedo López, Fundamentos de programación 2: tópicos de programación orientada a objetos, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021. ISBN: 978-612-5035-20-2. 170 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 20.5 x 29 cm.

	<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>

RUBRICA:

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
1. <u>GitHub</u>	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio <u>GitHub</u> con código fuente terminado y fácil de revisar.	2	<u>X</u>	2	
2. <u>Commits</u>	Hay capturas de pantalla de los <u>commits</u> más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	<u>X</u>	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	<u>X</u>	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	<u>X</u>	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	<u>X</u>	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	<u>X</u>	1	
7. Ortografía	El documento no muestra errores ortográficos.	2	<u>X</u>	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	<u>X</u>	3	
TOTAL		20		17	