INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Fundamentos de la Programación 02*				
TÍTULO DE LA PRÁCTICA:	*ArrayList*				
NÚMERO DE PRÁCTICA:	*06*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*2*
FECHA DE PRESENTACIÓN	*25/10/2024*	**HORA DE PRESENTACIÓN**	*17/30/00 PM*		
INTEGRANTE (s) *German Arturo Chipana Jerónimo*				**NOTA (0-20)**	
DOCENTE(s): *Pinto Oppe Lino José*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.

El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado

EJERCICIO 01:

1. Cree un Proyecto llamado Laboratorio6.

2. Usted deberá crear las dos clases Soldado.java y VideoJuego3.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. Pero ahora el tablero debe ser un ArrayList bidimensional.

5. Tendrá 2 Ejércitos. 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 (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.

MAIN:

```java
/*
 Autor : Chipana Jeronimo German Arturo
 Proposito : Laboratorio 06
*/
package laboratorio06_fp2;


public class Laboratorio06_FP2 {

    public static void main(String[] args) {
        VideoJuego3 juego=new VideoJuego3(10, 10);  // Crea una instancia del juego con un tablero de 10x10
        juego.mostrarTablero(); // Muestra el tablero actual con los soldados colocados
        juego.mostrarSoldadosMayorVida(1, juego.ejercito1); // Muestra el soldado con mayor nivel de vida del Ejército 1
        juego.mostrarSoldadosMayorVida(2, juego.ejercito2); // Muestra el soldado con mayor nivel de vida del Ejército 2
        juego.mostrarPromedioNivelVida(1, juego.ejercito1); // Calcula y muestra el promedio de nivel de vida del Ejército 1
        juego.mostrarPromedioNivelVida(2, juego.ejercito2); // Calcula y muestra el promedio de nivel de vida del Ejército 2
        juego.mostrarSoldadosOrdenCreacion(1, juego.ejercito1); // Muestra los soldados del Ejército 1 en el orden en que fueron creados
        juego.mostrarSoldadosOrdenCreacion(2, juego.ejercito2); // Muestra los soldados del Ejército 2 en el orden en que fueron creados
        juego.mostrarRankingSoldados(1, juego.ejercito1);   // Ordena y muestra el ranking de soldados del Ejército 1
        juego.mostrarRankingSoldados(2, juego.ejercito2);   // Ordena y muestra el ranking de soldados del Ejército 2
        juego.mostrarEjercitoGanador(juego.ejercito1, juego.ejercito2); // Determina y muestra el ejército ganador
    }
}
```

CLASE SOLDADO:

```java
/*
  Clase Soldado
 */
package laboratorio06_fp2;


public class Soldado {
    private String nombre;
    private int nivelVida;
    private int fila;
    private int columna;
    //Constructor
    public Soldado(String nombre, int nivelVida, int fila, int columna){
        this.nombre = nombre;
        this.nivelVida = nivelVida;
        this.fila = fila;
        this.columna = columna;
    }
    // Metodos accesores
    public String getNombre(){
        return nombre;
    }
    public int getNivelVida(){
        return nivelVida;
    }
    public int getFila(){
        return fila;
    }
    public int getColumna(){
        return columna;
    }
    // Metodo toString
    public String toString(){
        return "Soldado: "+nombre+", Vida: "+nivelVida+", Posicion: ("+(fila+1)+", "+(columna+1)+")";
    }
}
```

CLASE VIDEOJUEGO3:

```java
/*
  Clase VideoJuego3
*/
package laboratorio06_fp2;

import java.util.*;
public class VideoJuego3 {
    private ArrayList<ArrayList<Soldado>> tablero;  // Tablero bidimensional de soldados
    public ArrayList<Soldado> ejercito1;   // Lista de soldados del Ejército 1
    public ArrayList<Soldado> ejercito2;   // Lista de soldados del Ejército 2
    private int filas;
    private int columnas;
    // Constructor: Inicializa el tablero y los ejércitos
    public VideoJuego3(int filas, int columnas){
        this.tablero=new ArrayList<ArrayList<Soldado>>();   // Inicializa el tablero bidimensional
        this.ejercito1=new ArrayList<Soldado>();    // Inicializa la lista del Ejército 1
        this.ejercito2=new ArrayList<Soldado>();    // Inicializa la lista del Ejército 2
        this.filas=filas;
        this.columnas=columnas;
        inicializarTablero();   // Llena el tablero con espacios vacíos
        inicializarEjercito(1, ejercito1);  // Inicializa los soldados para el Ejército 1
        inicializarEjercito(2, ejercito2);  // Inicializa los soldados para el Ejército 2
    }
    // Método para inicializar el tablero con filas y columnas vacías
    private void inicializarTablero(){
        for(int i=0;i<filas;i++){
            tablero.add(new ArrayList<>()); // Agrega una nueva fila
            for(int j=0;j<columnas;j++){
                tablero.get(i).add(null);   // Llena cada celda con null para indicar que está vacía
            }
        }
    }
    // Método para inicializar los soldados en el tablero y añadirlos a un ejército
    private void inicializarEjercito(int numEjercito, ArrayList<Soldado> ejercito){
        int cantidadSoldados=(int)(Math.random()*10+1); // Genera una cantidad aleatoria de soldados entre 1 y 10
        for(int i=0;i<cantidadSoldados;i++){
            String nombre="Soldado"+i+"X"+numEjercito;  // Genera un nombre único para cada soldado
            int nivelVida=(int)(Math.random()*5+1);  // Asigna un nivel de vida aleatorio entre 1 y 5
            int fila, columna;
            // Busca una posición vacía en el tablero para colocar al soldado
            do{
                fila=(int)(Math.random()*10);   // Fila aleatoria
                columna=(int)(Math.random()*10);    // Columna aleatoria
            }while(tablero.get(fila).get(columna)!=null);   // Repite mientras la posición no esté vacía

            Soldado soldado=new Soldado(nombre, nivelVida, fila, columna);  // Crea el soldado con los atributos generados
            tablero.get(fila).set(columna, soldado);    // Coloca el soldado en el tablero
            ejercito.add(soldado);  // Añade el soldado al ejército correspondiente
        }
    }
```

```java
    // Método para mostrar el tablero con los soldados posicionados
    public void mostrarTablero(){
        for(int i=0;i<filas;i++){
            for(int j=0;j<columnas;j++){
                Soldado soldado=tablero.get(i).get(j);  // Obtiene el soldado en la posición (i,j)
                if(soldado==null){
                    System.out.print("| _________ ");   // Imprime una casilla vacía
                }else{
                    System.out.print("|"+soldado.getNombre()+" ");  // Imprime el nombre del soldado en la casilla
                }
            }
            System.out.println("|");    // Nueva línea después de cada fila
        }
        System.out.println();
    }
    // Método para mostrar el soldado con mayor nivel de vida en un ejército
    public void mostrarSoldadosMayorVida(int numEjercito, ArrayList<Soldado> ejercito){
        Soldado mayorVida=ejercito.get(0);  // Inicializa con el primer soldado
        for(Soldado soldado : ejercito){
            if(soldado.getNivelVida()>mayorVida.getNivelVida()){
                mayorVida=soldado;  // Actualiza si encuentra un soldado con mayor nivel de vida
            }
        }
        System.out.println("Soldado con mayor nivel de vida del ejercito "+numEjercito+" : "+mayorVida);
        System.out.println();
    }
    // Método para calcular y mostrar el promedio del nivel de vida de un ejército
    public void mostrarPromedioNivelVida(int numEjercito, ArrayList<Soldado> ejercito){
        int sumaVida=0;
        for(Soldado soldado : ejercito){
            sumaVida+=soldado.getNivelVida();   // Suma el nivel de vida de cada soldado
        }
        double promedio=(double)sumaVida/ejercito.size();   // Calcula el promedio
        System.out.println("Promedio del nivel de vida del ejercito "+numEjercito+" : "+promedio);
        System.out.println();
    }
    // Método para mostrar los soldados en el orden en que fueron creados
    public void mostrarSoldadosOrdenCreacion(int numEjercito, ArrayList<Soldado> ejercito){
        System.out.println("Soldados del ejercito "+numEjercito+" en el orden de creacion:");
        for(Soldado soldado : ejercito){
            System.out.println(soldado);    // Imprime cada soldado en el orden de creación
        }
        System.out.println();
    }
```

```java
    // Método para mostrar el ranking de soldados basado en su nivel de vida, usando burbuja y selección
    public void mostrarRankingSoldados(int numEjercito, ArrayList<Soldado> ejercito){
        // Ordenamiento por burbuja
        ArrayList<Soldado> ejercitoBurbuja=(ArrayList<Soldado>) ejercito.clone();   // Clona el ejército original
        burbujaOrdenar(ejercitoBurbuja);    // Ordena usando el algoritmo de burbuja
        System.out.println("Ranking de soldados ejercito "+numEjercito+" (Burbuja): ");
        for(Soldado soldado : ejercitoBurbuja){
            System.out.println(soldado);    // Imprime los soldados ordenados
        }
        // Ordenamiento por selección
        ArrayList<Soldado> ejercitoSeleccion=(ArrayList<Soldado>) ejercito.clone(); // Clona el ejército original
        seleccionOrdenar(ejercitoSeleccion);    // Ordena usando el algoritmo de selección
        System.out.println("Ranking de soldados ejercito "+numEjercito+" (Seleccion): ");
        for(Soldado soldado : ejercitoSeleccion){
            System.out.println(soldado);    // Imprime los soldados ordenados
        }
        System.out.println();
    }
    // Método de ordenamiento por burbuja para ordenar soldados por nivel de vida (descendente)
    private void burbujaOrdenar(ArrayList<Soldado> soldados){
        for(int i=0;i<soldados.size()-1;i++){
            for(int j=0;j<soldados.size()-1-i;j++){
                if(soldados.get(j).getNivelVida()<soldados.get(j+1).getNivelVida()){
                    // Intercambia los soldados si el nivel de vida del siguiente es mayor
                    Soldado temp=soldados.get(j);
                    soldados.set(j, soldados.get(j+1));
                    soldados.set(j+1, temp);
                }
            }
        }
    }
    // Método de ordenamiento por selección para ordenar soldados por nivel de vida (descendente)
    private void seleccionOrdenar(ArrayList<Soldado> soldados){
        for(int i=0;i<soldados.size()-1;i++){
            int maxIndex=i; // Asume que el soldado en la posición i tiene el mayor nivel de vida
            for(int j=i+1;j<soldados.size();j++){
                if(soldados.get(j).getNivelVida()>soldados.get(maxIndex).getNivelVida()){
                    maxIndex=j; // Actualiza la posición del soldado con mayor nivel de vida
                }
            }
            // Intercambia el soldado en i con el de mayor nivel de vida encontrado
            Soldado temp=soldados.get(i);
            soldados.set(i, soldados.get(maxIndex));
            soldados.set(maxIndex, temp);
        }
    }
```

```java
    // Método que determina y muestra el ejército ganador basado en el número de soldados
    public void mostrarEjercitoGanador(ArrayList<Soldado> ejercito1, ArrayList<Soldado> ejercito2){
        if(ejercito1.size()>=ejercito2.size()){
            if(ejercito1.size()==ejercito2.size())
                System.out.println("Hubo un EMPATE de ejercitos, ambos con "+ejercito1.size()+" soldados...");
            else
                System.out.println("Gano el EJERCITO 1 con "+ejercito1.size()+" soldados!");
        }
        else
            System.out.println("Gano el EJERCITO 2 con "+ejercito2.size()+" soldados!");
    }
}
```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

- **Tamaño del tablero**: Comprobé con un tablero de 10x10 filas y columnas.
- **Cantidad de soldados**: La cantidad de soldados por ejército varía de 1 a 10 de manera aleatoria.

¿Qué resultado esperabas obtener para cada valor de entrada?

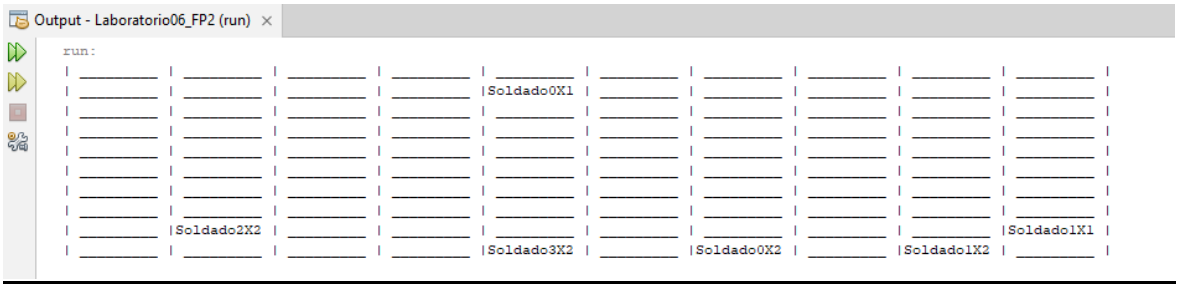
- **Tablero**: Esperaba ver un tablero de 10x10 con algunas casillas llenas de soldados (representados por nombres como "Soldado0X1") y otras vacías (con null).
- **Mayor nivel de vida**: Esperaba que se identificara correctamente el soldado con el mayor nivel de vida (valor entre 1 y 5) para cada ejército.
- **Promedio de nivel de vida**: Esperaba obtener un número decimal que reflejara el promedio del nivel de vida de los soldados de cada ejército.
- **Orden de creación**: Esperaba ver los soldados de cada ejército impresos en el orden en que fueron generados, con su nombre y nivel de vida.
- **Ranking de soldados**: Esperaba ver los soldados ordenados correctamente en dos rankings (burbuja y selección), ambos descendiendo por el nivel de vida.
- **Ejército ganador**: Esperaba que el ejército con más soldados fuera declarado ganador, o que se indicara un empate si ambos ejércitos tenían el mismo número de soldados.

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

- **Tablero**: Obtuve un tablero de 10x10, donde varias posiciones estaban ocupadas por soldados y otras vacías, como se esperaba.
- **Mayor nivel de vida**: El código identificó correctamente al soldado con el nivel más alto de vida para cada ejército.
- **Promedio de nivel de vida**: El promedio calculado para cada ejército fue correcto y reflejó los niveles de vida de los soldados creados aleatoriamente.
- **Orden de creación**: Los soldados fueron mostrados en el orden en que fueron creados, con su nombre, nivel de vida y posición en el tablero.
- **Ranking de soldados**: Los rankings obtenidos por los algoritmos de burbuja y selección coincidieron y ordenaron a los soldados de mayor a menor nivel de vida, como se esperaba.
- **Ejército ganador**: El ejército con más soldados fue declarado ganador, o se mostró un mensaje de empate si tenían la misma cantidad de soldados.

LABORATORIO_06_FP2

1. Se mostró el tablero con los soldados de cada Ejército en su respectiva posición.



2. Se mostró el soldado con mayor nivel de vida el Ejército 1.

```
Soldado con mayor nivel de vida del ejercito 1 : Soldado: Soldado0X1, Vida: 5, Posicion: (2, 5)
```

3. Se mostró el soldado con mayor nivel de vida el Ejército 2.

```
Soldado con mayor nivel de vida del ejercito 2 : Soldado: Soldado0X2, Vida: 2, Posicion: (10, 7)
```

4. Se mostró el promedio del nivel de vida de todos los soldados del Ejército 1.

```
Promedio del nivel de vida del ejercito 1 : 4.0
```

5. Se mostró el promedio del nivel de vida de todos los soldados del Ejército 2.

```
Promedio del nivel de vida del ejercito 2 : 1.5
```

6. Se mostró a todos los soldados creados del Ejército 1 por orden de creación.

```
Soldados del ejercito 1 en el orden de creacion:
Soldado: Soldado0X1, Vida: 5, Posicion: (2, 5)
Soldado: Soldado1X1, Vida: 3, Posicion: (9, 10)
```

7. Se mostró a todos los soldados creados del Ejército 1 por orden de creación.

```
Soldados del ejercito 2 en el orden de creacion:
Soldado: Soldado0X2, Vida: 2, Posicion: (10, 7)
Soldado: Soldado1X2, Vida: 1, Posicion: (10, 9)
Soldado: Soldado2X2, Vida: 1, Posicion: (9, 2)
Soldado: Soldado3X2, Vida: 2, Posicion: (10, 5)
```

8. Se ordenaron los soldados del Ejército 1 por nivel de vida usando el ORDENAMIENTO BURBUJA y ORDENAMIENTO SELECCIÓN.

```
Ranking de soldados ejercito 1 (Burbuja):
Soldado: Soldado0X1, Vida: 5, Posicion: (2, 5)
Soldado: Soldado1X1, Vida: 3, Posicion: (9, 10)
Ranking de soldados ejercito 1 (Seleccion):
Soldado: Soldado0X1, Vida: 5, Posicion: (2, 5)
Soldado: Soldado1X1, Vida: 3, Posicion: (9, 10)
```

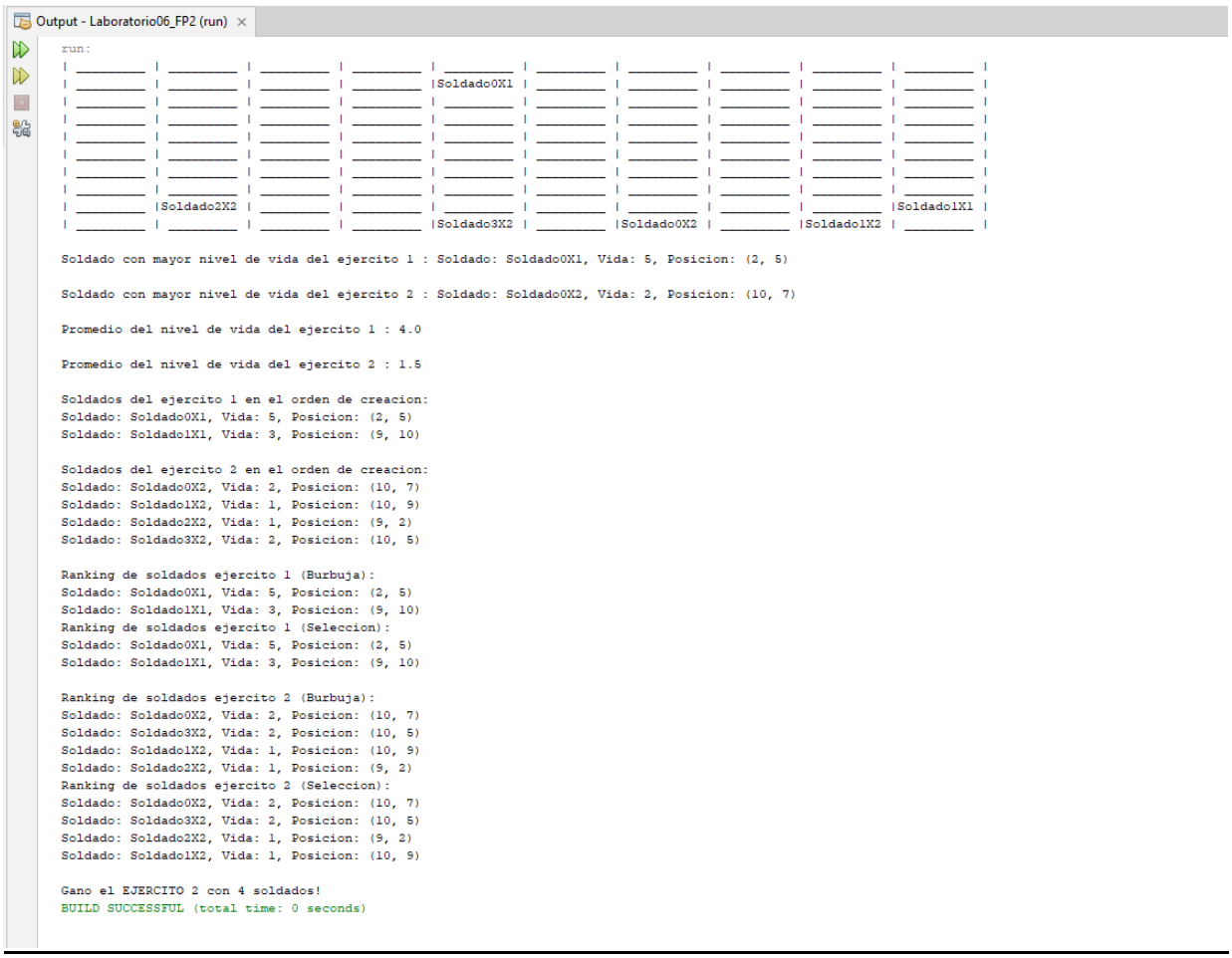
9. Se ordenaron los soldados del Ejército 2 por nivel de vida usando el ORDENAMIENTO BURBUJA y ORDENAMIENTO SELECCIÓN.

```
Ranking de soldados ejercito 2 (Burbuja):
Soldado: Soldado0X2, Vida: 2, Posicion: (10, 7)
Soldado: Soldado3X2, Vida: 2, Posicion: (10, 5)
Soldado: Soldado1X2, Vida: 1, Posicion: (10, 9)
Soldado: Soldado2X2, Vida: 1, Posicion: (9, 2)
Ranking de soldados ejercito 2 (Seleccion):
Soldado: Soldado0X2, Vida: 2, Posicion: (10, 7)
Soldado: Soldado3X2, Vida: 2, Posicion: (10, 5)
Soldado: Soldado2X2, Vida: 1, Posicion: (9, 2)
Soldado: Soldado1X2, Vida: 1, Posicion: (10, 9)
```

10. Se mostró que Ejército ganó la batalla y con cuantos soldados.

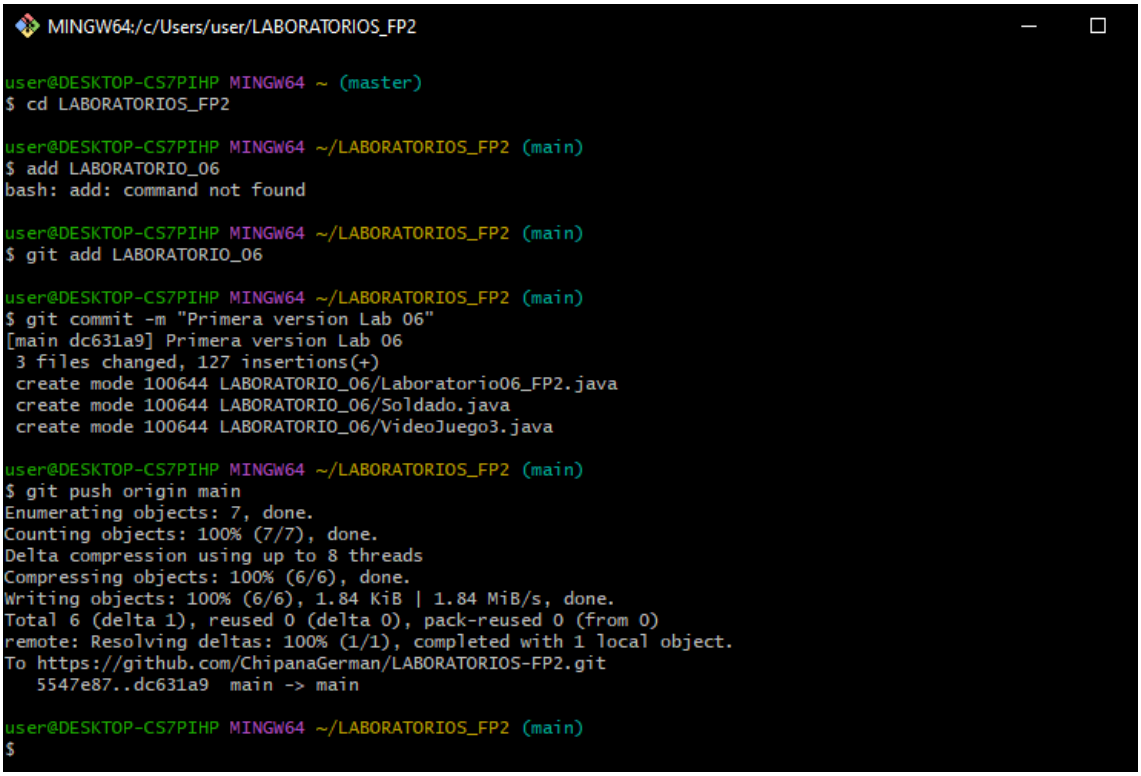
```
Gano el EJERCITO 2 con 4 soldados!
BUILD SUCCESSFUL (total time: 0 seconds)
```

11. Ejecución completa.



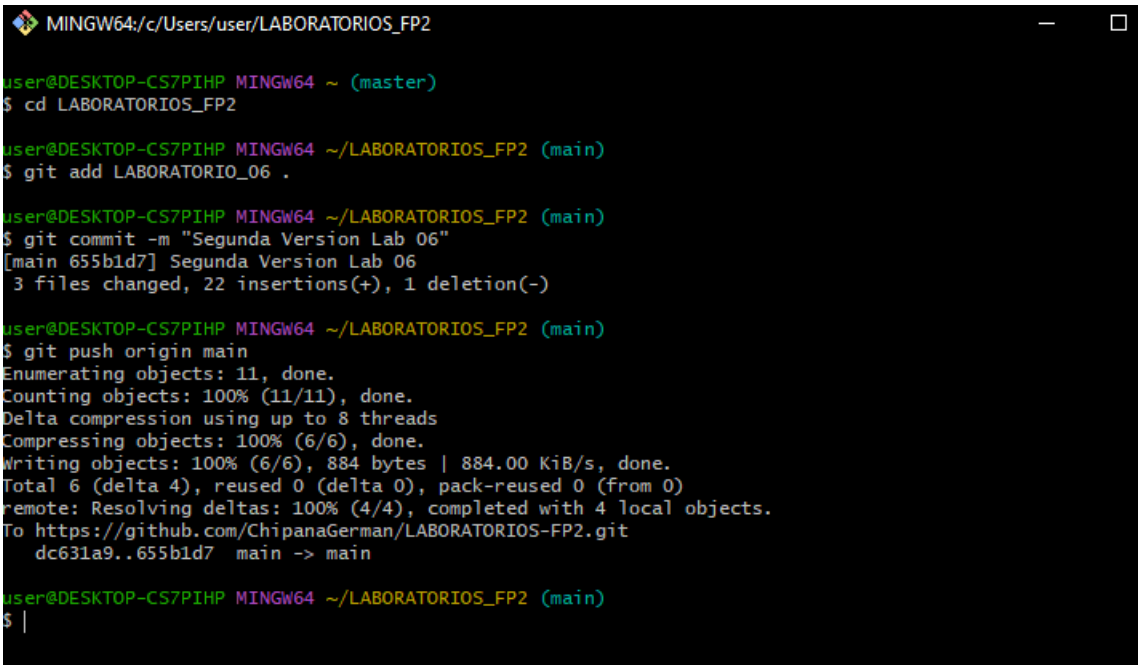
MIS COMMITS:

PRIMERA VERSION:



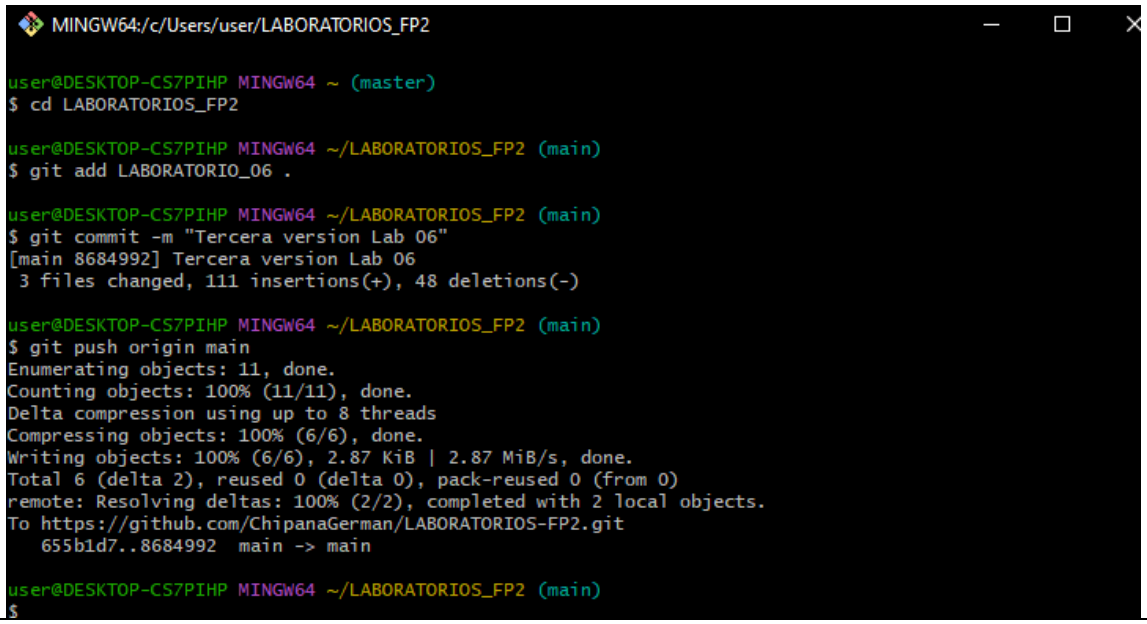
Se agregaron los archivos Laboratorio_06_FP2.java, Soldado.java y VideoJuego3.java al repositorio de GitHub.

SEGUNDA VERSION:



Se actualizaron los archivos Laboratorio_06_FP2.java, Soldado.java y VideoJuego3.java y se subieron a GitHub.

TERCERA VERSION:



Se actualizaron los archivos Laboratorio_06_FP2.java, Soldado.java y VideoJuego3.java y se subieron a GitHub.

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	X	1	
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	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 están 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	4	X	4	

	evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).				
TOTAL		20		18	

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

CONCLUSIONES

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

CONCLUSIÓN:

En esta práctica de laboratorio se implementaron con éxito ArrayLists bidimensionales para gestionar un tablero de juego y los soldados en él. Se trabajaron de manera eficiente los algoritmos de ordenamiento por burbuja y selección**,** lo que permitió ordenar los soldados según su nivel de vida. La práctica ayudó a reforzar el uso de estructuras dinámicas y algoritmos fundamentales, aplicados a la resolución de problemas prácticos en Java.

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.

- **Análisis del problema**: Se revisaron los requisitos de la práctica, comprendiendo la necesidad de implementar un tablero bidimensional y trabajar con soldados distribuidos aleatoriamente.
- **Diseño de clases**: Se definieron las clases Soldado y VideoJuego3 para estructurar los objetos soldados y gestionar el tablero y las operaciones requeridas.
- **Implementación del código**: Se programaron las funcionalidades clave, incluyendo la inicialización del tablero, la creación aleatoria de soldados, y los métodos de ordenamiento por burbuja y selección.

- **Pruebas y validación**: Se realizaron pruebas ejecutando el juego con diferentes tamaños de tablero y cantidades de soldados, verificando el correcto posicionamiento de los soldados y los cálculos de vida.
- **Depuración**: Se corrigieron errores encontrados durante las pruebas, optimizando el código y ajustando el manejo de posiciones vacías en el tablero.
- **Documentación**: Finalmente, se documentó el código y las versiones fueron subidas a un repositorio en GitHub para su revisión y control de versiones.

REFERENCIAS Y BIBLIOGRAFÍA

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

M. Aedo López, Práctica de Laboratorio 6: ArrayList, Universidad Nacional de San Agustín, 2023. https://github.com/ChipanaGerman/LABORATORIOS-FP2