

Informe de Laboratorio 08

Tema: Laboratorio 08

Nota

Estudiante	Escuela	Asignatura
Victor Mamani Anahua vmamanian@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación II Semestre: II Código: 20230489

Laboratorio	Tema	Duración
08	Laboratorio 08	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 17 Octubre 2023	Al 24 Octubre 2023

1. Tarea

- Cree un Proyecto llamado Laboratorio8
- Usted deberá crear las dos clases Soldado.java y VideoJuego5.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.
- Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada.
- Tendrá 2 Ejércitos (usar HashMaps). 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).
- 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 (indicar conclusiones respecto a este ordenamiento de HashMaps).
- Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla).

2. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 23 lunar 64 bits Kernell 6.2.v
- Visual Studio Code.
- VIM 9.0.
- OpenJDK 64-Bits 19.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Actividades del Laboratorio 08.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/VictorMA18/fp2-23b.git`
- URL para el laboratorio 08 en el Repositorio GitHub.
- `https://github.com/VictorMA18/fp2-23b/tree/main/Fase02/Lab08`

4. Actividades del Laboratorio 08

4.1. Ejercicio Soldado

- En el primer commit bueno reutilizamos el archivo que seria nuestra clase Soldado el cual la utilizaremos para poder avanzar con el siguiente ejercicio que seria VideoJuego5.
- El codigo y el commit seria el siguiente:

Listing 1: Commit

```
$ git commit -m "Publicando la clase Soldado para su posterior uso en el siguiente  
ejercicio que es con Hashmaps"
```

Listing 2: Las lineas de codigos del metodo creado:

```
// Laboratorio Nro 8 - Ejercicio Soldado  
// Autor: Mamani Anahua Victor Narciso  
// Colaboro:  
// Tiempo:  
public class Soldado { //CREAMOS LA CLASE SOLDODADO PARA PODER USAR UN ARREGLO  
    BIDIMENSIONAL DONDE NECESITAMOS LA VIDA , EL NOMBRE DEL SOLDADO Y TAMBIEN SU  
    POSICION COMO LA FILA Y LA COLUMNA  
  
    private String name;  
    private int heatlh;  
    private int row;  
    private String column;
```

```
//Constructor
public Soldado(String name, int health, int row, String column){
    this.name = name;
    this.health = health;
    this.row = row;
    this.column = column;
}

// Metodos mutadores
public void setName(String n){
    name = n;
}
public void setHealth(int p){
    health = p;
}
public void setRow(int b){
    row = b;
}
public void setColumn(String c){
    column = c;
}

// Metodos accesorios
public String getName(){
    return name;
}
public int getHealth(){
    return health;
}

public int getRow(){
    return row;
}
public String getColumn(){
    return column;
}

// Completar con otros metodos necesarios
public String toString(){ //CREAMOS ESTE METODO PARA IMPRIMIR LOS DATOS DEL OBJETO
    String join = "\nNombre: " + getName() + "\nVida: " + getHealth() + "\nFila: " +
        getRow() + "\nColumna: " + getColumn(); //Agregamos un espaciador para poder
        separar
    return join;
}
}
```

4.2. Ejercicio VideoJuego5

- En el segundo commit creamos el metodo `mapHashFillRegister()` el cual usamos el uso de Hash-Maps y tambien de `ArrayList` para poder añadirlos al hashmaps ya que en este haríamos la comprobacion de que cada Soldado gracias a una iteracion el cual este comprueba que estos si son nulos se añadiría al soldado en la cuadrilla correspondiente y si no lo añadiría y retrocedería una iteracion ya que se contaría por lo que nos lleva a que en este no se repita en misma cuadrilla despues de esto imprimiríamos los datos del soldado en el orden que fueron creados

Listing 3: Commit

```
$ git commit -m "Creando el metodo mapHashFillRegister() el cual usamos el uso de  
HashMaps y tambien de ArrayList para poder anadirlos al hashmaps ya que en este  
hariaos la comprobacion de que cada Soldado en este no se repita en misma  
cuadrilla despues de esto imprimiriamos los datos del soldado en el orden que  
fueron creados"
```

Listing 4: Las lineas de codigos del metodo creado:

```
public static void viewBoard(HashMap<String, Soldado> army1, HashMap<String, Soldado>  
    army2){ //EN ESTE METODO DEMOSTRAREMOS LA TABLA REUTILIZAREMOS CODIGOS DE  
    ANTERIORES LABORATORIOS PARA PODER HACER LA BASE DE ESTE TABLERO  
    System.out.println("\nMostrando tabla de posicion --");  
    System.out.println("Leyenda: Ejercito1 --> X | Ejercito2 --> Y"); //RECONOCIMIENTO  
    PARA LOS EJERCITOS Y POSICION DE SUS SOLDADOS  
    System.out.println("\n \t A\t B\t C\t D\t E\t F\t G\t H\t I\t J"); //  
    RECONOCIMIENTO PARA CADA UBICACION DE CADA SOLDADO EN EL TABLERO POR PARTE DE  
    LAS COLUMNAS  
    System.out.println("\t-----");  
    for(int i = 0; i < 10; i++){  
        System.out.print((i + 1) + "\t"); // RECONOCIMIENTO PARA CADA UBICACION DE CADA  
        SOLDADO EN EL TABLERO POR PARTE DE LAS FILAS  
        for(int j = 0; j < 10; j++){  
            if(army1.get("Soldado" + i + "X" + j) != null){  
                System.out.print("| " + "X" + " "); //VERIFICANDOLA POSICIONES DE CADA  
                SOLDADO DE CADA EJERCITO CON SU RESPECTIVO INDICADOR PARA PODER  
                UBICARLOS  
            }else if(army2.get("Soldado" + i + "X" + j) != null){  
                System.out.print("| " + "Y" + " ");  
            }else{  
                System.out.print("|   ");  
            }  
        }  
        System.out.println("|");  
        System.out.println("\t|-----|-----|-----|-----|-----|-----|-----|-----|");  
    }  
    System.out.println("\n*****");  
}
```

Listing 5: Ejecucion:

```
El Ejercito 1 tiene 5 soldados :  
*****  
Registrando al 1 soldado del Ejercito 1  
-----  
Nombre: Soldado0X1  
Vida: 5  
Fila: 9  
Columna: C  
Registrando al 2 soldado del Ejercito 1  
-----  
Nombre: Soldado1X1  
Vida: 1  
Fila: 10
```

```
Columna: H
Registrando al 3 soldado del Ejercito 1
-----
Nombre: Soldado2X1
Vida: 3
Fila: 1
Columna: F
Registrando al 4 soldado del Ejercito 1
-----
Nombre: Soldado3X1
Vida: 1
Fila: 1
Columna: I
Registrando al 5 soldado del Ejercito 1
-----
Nombre: Soldado4X1
Vida: 2
Fila: 8
Columna: I
El Ejercito 2 tiene 2 soldados :
*****
Registrando al 1 soldado del Ejercito 2
-----
Nombre: Soldado0X2
Vida: 4
Fila: 8
Columna: B
Registrando al 2 soldado del Ejercito 2
-----
Nombre: Soldado1X2
Vida: 4
Fila: 9
Columna: F

Mostrando tabla de posicion ... --
Leyenda: Ejercito1 --> X | Ejercito2 --> Y
```

4.3. Ejercicio VideoJuego5

- En el tercer commit creamos el metodo viewBoard() el cual nos va poder ayudar a visualizar el tablero el cual necesitamos saber que dentro del Hashmap se encuentre un soldado que no sea nulo el cual respectivamente en el bando que se encuentre le dara su respectivo reconocimiento para esto modificamos tambien el nombre de la clave en el hashmap el cual este nos dice manera mas clara la posicion de este soldado

Listing 6: Commit

```
$ git commit -m "Creamos el metodo viewBoard() el cual nos va poder ayudar a visualizar
el tablero el cual necesitamos saber que dentro del Hashmap se encuentre un soldado
que no sea nulo el cual respectivamente en el bando que se encuentre le dara su
respectivo reconocimiento para esto modificamos tambien el nombre de la clave en el
hashmap el cual este nos dice manera mas clara la posicion de este soldado"
```

Listing 7: Las líneas de códigos del método creado:

```
// Laboratorio Nro 8 - Ejercicio VideoJuego5
// Autor: Mamani Anahua Victor Narciso
// Colaboro:
// Tiempo:
import java.util.*;
class VideoJuego5{
    public static HashMap<String, Soldado> mapHashFillRegister(int num){
        Random rdm =new Random();
        HashMap<String, Soldado> army1 = new HashMap<String, Soldado>();
        ArrayList<ArrayList<Soldado>> army = new ArrayList<ArrayList<Soldado>>(); //NOS
        AYUDARIAMOS DE UN ARRAYLIST PARA PODER AYUDARNOS CON EL USO DE HASHMAPS PARA
        PODER REGISTRAR A SOLDADOS EN LA QUE NINGUNO DE ESTOS SE REPITA
        int numsoldiers = rdm.nextInt(10) + 1; //NUMERO DE SOLDADOS QUE SE VAN A CREAR DE
        1 AL 10
        for(int i = 0; i < 10; i++){
            army.add(new ArrayList<Soldado>()); //LLENAMOS NUESTROS ARRAYLIST BIDIMENSIONAL
            CON CADA FILA PARA QUE CUMPLAN CON ESTRUCTURA DEL TABLERO
            for(int j = 0; j < 10 ; j++){
                army.get(i).add(null); // LLENAMOS CADA FILA DEL ARRAYLIST CON UN OBJETO
                SOLDADO CON TAL QUE ESTE SEA NULL PARA QUE SEPA QUE ESTE TIENE UNA
                CASILLA PERO NO HAY NADIE TODAVIA SE PUEDE LLENAR
            }
        }
        System.out.println("El Ejercito " + num + " tiene " + numsoldiers + " soldados : "
        );
        System.out.println("*****");
        for(int i = 0; i < numsoldiers; i++){ //ITERACION PARA PODER DARLES LOS DATOS A
        CADA SOLDADO CREADO
            String name = "Soldado" + i + "X" + num;
            int health = rdm.nextInt(5) + 1;
            int row = rdm.nextInt(10) + 1;
            String column = String.valueOf((char)(rdm.nextInt(10) + 65)); //REUTILIZAMOS
            CODIGO DEL ANTERIOR ARCHIVO VIDEOJUEGO4.JAVA YA QUE TENDRIAN LA MISMA
            FUNCIONALIDAD
            if(army.get(row - 1).get((int)column.charAt(0) - 65) == null){
                System.out.println("Registrando al " + (i + 1) + " soldado del Ejercito " +
                num + "");
                System.out.print("-----");
                army.get(row - 1).set((int)column.charAt(0) - 65, new Soldado(name, health,
                row, column));
                army1.put("Soldado" + i, new Soldado(name, health, row, column));
                //INTEGRAMOS AL HASHMAP AL SOLDADO CON SU RESPECTIVO NOMBRE Y VALOR
                System.out.println(army1.get("Soldado"+ i).toString()); //PUBLICAMOS AL
                SOLDADO CREADO POR ORDEN DE CREACION
            }else{
                i -= 1; //NOS AYUDARIA CON LOS SOLDADOS QUE SE REPITEN EN EL MISMO CASILLERO
                CON TAL QUE NO DEBERIA CONTAR
            }
        }
        return army1;
    }
    public static void main (String args []){
        HashMap<String, Soldado> army1 = mapHashFillRegister(1);
        HashMap<String, Soldado> army2 = mapHashFillRegister(2);
    }
}
```

```
}
```

Listing 8: Ejecucion:

```
El Ejercito 1 tiene 4 soldados :
*****
Registrando al 1 soldado del Ejercito 1
-----
Nombre: Soldado0X1
Vida: 5
Fila: 4
Columna: D
Registrando al 2 soldado del Ejercito 1
-----
Nombre: Soldado1X1
Vida: 3
Fila: 8
Columna: C
Registrando al 3 soldado del Ejercito 1
-----
Nombre: Soldado2X1
Vida: 5
Fila: 6
Columna: C
Registrando al 4 soldado del Ejercito 1
-----
Nombre: Soldado3X1
Vida: 5
Fila: 8
Columna: I
El Ejercito 2 tiene 1 soldados :
*****
Registrando al 1 soldado del Ejercito 2
-----
Nombre: Soldado0X2
Vida: 5
Fila: 2
Columna: B

Mostrando tabla de posicion ... --
Leyenda: Ejercito1 --> X | Ejercito2 --> Y
```

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

4.4. Ejercicio VideoJuego5

- En el cuarto commit modificamos el metodo viewBoard() en este haremos comparaciones entre los 2 ejercitos debido que en este podemos ver que 1 soldado de cada ejercito se puede cruzar debido a esto haremos una comprobacion de estos soldado de quien tiene mas vida para que se que de con ese casillero para eso va a ver una batalla la cual el ganador saldra con menos vida debido a la batalla y ganara el casillero para esto usamos bastantes if para comprobar esta situacion y vamos iterando por cada casillero para ver cada caso

Listing 9: Commit

```
$ git commit -m "Arreglando el tablero con las posiciones de cada bando de cada
ejercito en caso de que soldados de diferente ejercito se encuentren en el mismo
casillero estos van a tener una batalla como vemos en la que se compara su nivel de
vida y el que tenga mas vida se posicionara donde esta siendo este el ganador pero
su vida se reducira debido laa batalla tenida con el otro soldado el cual va a ser
eliminado del campo"
```

Listing 10: Las lineas de codigos del metodo creado:

```
public static void viewBoard(HashMap<String, Soldado> army1, HashMap<String, Soldado>
    army2){ //EN ESTE METODO DEMOSTRAREMOS LA TABLA REUTILIZAREMOS CODIGOS DE
    ANTERIORES LABORATORIOS PARA PODER HACER LA BASE DE ESTE TABLERO
    System.out.println("\nMostrando tabla de posicion ... --");
    System.out.println("Leyenda: Ejercito1 --> X | Ejercito2 --> Y"); //RECONOCIMIENTO
    PARA LOS EJERCITOS Y POSICION DE SUS SOLDADOS
    System.out.println("\n \t A\t B\t C\t D\t E\t F\t G\t H\t I\t J"); //
    RECONOCIMIENTO PARA CADA UBICACION DE CADA SOLDADO EN EL TABLERO POR PARTE DE
    LAS COLUMNAS
    System.out.println("\t_-----");
    for(int i = 0; i < 10; i++){
        System.out.print((i + 1) + "\t"); // RECONOCIMIENTO PARA CADA UBICACION DE CADA
        SOLDADO EN EL TABLERO POR PARTE DE LAS FILAS
```


Listing 11: Ejecucion:

Página 9

```
-----
Nombre: Soldado2X1
Vida: 1
Fila: 3
Columna: A
Registrando al 4 soldado del Ejercito 1
-----
Nombre: Soldado3X1
Vida: 5
Fila: 2
Columna: D
El Ejercito 2 tiene 4 soldados :
*****
Registrando al 1 soldado del Ejercito 2
-----
Nombre: Soldado0X2
Vida: 5
Fila: 2
Columna: J
Registrando al 2 soldado del Ejercito 2
-----
Nombre: Soldado1X2
Vida: 1
Fila: 7
Columna: J
Registrando al 3 soldado del Ejercito 2
-----
Nombre: Soldado2X2
Vida: 1
Fila: 9
Columna: B
Registrando al 4 soldado del Ejercito 2
-----
Nombre: Soldado3X2
Vida: 2
Fila: 6
Columna: J

Mostrando tabla de posicion ... --
Leyenda: Ejercito1 --> X | Ejercito2 --> Y
```

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

4.5. Ejercicio VideoJuego5

- En el quinto commit creamos el metodo longerLife() el cual nos podra a dar la informacion del soldado el cual tenga la mayor vida este se va comparando con otros soldados el cual tendra que hacer una iteracion con cada soldado de cada ejercito a la vez con esto comprobamos que el soldado el cual estamos buscando no sea nulo

Listing 12: Commit

```
$ git commit -m "Creamos el metodo longerLife() el cual nos podra a dar la informacion del soldado el cual tenga la mayor vida este se va comparando con otros soldados el cual tendra que hacer una iteracion con cada soldado de cada ejercito a la vez con esto comprobamos que el soldado el cual estamos buscando no sea nulo"
```

Listing 13: Las lineas de codigos del metodo creado:

```
public static void longerLife(HashMap<String, Soldado> army , int num){
    int mayor = 0;
    Soldado soldier = null;
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < 10; j++){ //ITERACION LA CUAL NOS AYUDA A PASAR POR TODOS LOS SOLDADOS DE CADA EJERCITO
            if(army.get("Soldado" + i + "X" + j) != null){ //VERIFICAMOS QUE EL SOLDADO EL CUAL ESTAMOS VERIFICANOD NO SEA UNO NULO
                if(army.get("Soldado" + i + "X" + j).getHealth() > mayor){
                    mayor = army.get("Soldado" + i + "X" + j).getHealth(); //DETECTAMOS EL MAYOR EL CUAL VAMOS COMPRANDO CON LOS DEMAS SOLDADOS PARA TENER SOLO AL SOLDADO EL CUAL TENGA LA MAYOR VIDA
                    soldier = army.get("Soldado" + i + "X" + j); //SOLDIER CONTENDRA A ESTE SOLDADO EL CUAL DESPUES SE IMPRIMIRA SUS DATOS PARA VER DE QUE SOLDADO SE TRATA
                }
            }
        }
    }
}
```

```
    }  
  }  
  System.out.println("");  
  System.out.println("El soldado con mayor vida del Ejercito " + num + " es: ");  
  System.out.println(soldier.toString());  
  System.out.println("*****");  
}
```

Listing 14: Ejecucion:

```
El Ejercito 1 tiene 7 soldados :  
*****  
Registrando al 1 soldado del Ejercito 1  
-----  
Nombre: Soldado0X1  
Vida: 5  
Fila: 9  
Columna: I  
Registrando al 2 soldado del Ejercito 1  
-----  
Nombre: Soldado1X1  
Vida: 1  
Fila: 3  
Columna: D  
Registrando al 3 soldado del Ejercito 1  
-----  
Nombre: Soldado2X1  
Vida: 5  
Fila: 9  
Columna: E  
Registrando al 4 soldado del Ejercito 1  
-----  
Nombre: Soldado3X1  
Vida: 5  
Fila: 4  
Columna: G  
Registrando al 5 soldado del Ejercito 1  
-----  
Nombre: Soldado4X1  
Vida: 4  
Fila: 3  
Columna: C  
Registrando al 6 soldado del Ejercito 1  
-----  
Nombre: Soldado5X1  
Vida: 1  
Fila: 2  
Columna: C  
Registrando al 7 soldado del Ejercito 1  
-----  
Nombre: Soldado6X1  
Vida: 3  
Fila: 10  
Columna: J  
El Ejercito 2 tiene 9 soldados :  
*****
```

```
Registrando al 1 soldado del Ejercito 2
-----
Nombre: Soldado0X2
Vida: 3
Fila: 1
Columna: H
Registrando al 2 soldado del Ejercito 2
-----
Nombre: Soldado1X2
Vida: 3
Fila: 2
Columna: A
Registrando al 3 soldado del Ejercito 2
-----
Nombre: Soldado2X2
Vida: 4
Fila: 10
Columna: G
Registrando al 4 soldado del Ejercito 2
-----
Nombre: Soldado3X2
Vida: 3
Fila: 9
Columna: H
Registrando al 5 soldado del Ejercito 2
-----
Nombre: Soldado4X2
Vida: 4
Fila: 6
Columna: H
Registrando al 6 soldado del Ejercito 2
-----
Nombre: Soldado5X2
Vida: 4
Fila: 9
Columna: F
Registrando al 7 soldado del Ejercito 2
-----
Nombre: Soldado6X2
Vida: 4
Fila: 4
Columna: A
Registrando al 8 soldado del Ejercito 2
-----
Nombre: Soldado7X2
Vida: 2
Fila: 9
Columna: E
Registrando al 9 soldado del Ejercito 2
-----
Nombre: Soldado8X2
Vida: 5
Fila: 6
Columna: J

Mostrando tabla de posicion ... --
```

Leyenda: Ejercito1 --> X | Ejercito2 --> Y

	A	B	C	D	E	F	G	H	I	J
1	Informe08.pdf							Y		
2	Y		X							
3	Soldado.java		X	X						
4	Y						X			
5										
6								Y		Y
7										
8										
9					X	Y		Y	X	
10							Y			X

Listing 15: Ejecucion:

```
*****
El soldado con mayor vida del Ejercito 1 es:

Nombre: Soldado3X1
Vida: 5
Fila: 4
Columna: G
*****

El soldado con mayor vida del Ejercito 2 es:

Nombre: Soldado8X2
Vida: 5
Fila: 6
Columna: J
*****
```

4.6. Ejercicio VideoJuego5

- En el sexto commit creamos el metodo `averageLife()` en este metodo recolectamos la vida de cada uno de los soldados de un ejercito el cual se va recolectando para despues dividirlo entre la cantidad de soldados que hay en el ejercito esto nos devolviera este promedio el cual tambien se publica para el conocimiento del usuario

Listing 16: Commit

```
$ git commit -m "Creando el metodo averageLife() en este metodo recolectamos la vida de cada uno de los soldados de un ejercito el cual se va recolectando para despues dividirlo entre la cantidad de soldados que hay en el ejercito esto nos devolvera este promedio el cual tambien se publica para el conocimiento del usuario"
```

Listing 17: Las lineas de codigos del metodo creado:

```
public static double averageLife(HashMap<String, Soldado> army , int num){  
    int sum = 0;  
    int count = 0;  
    Soldado soldier = null;  
    System.out.println("El promedio de puntos de vida del Ejercito " + num + " es: ");  
    for(int i = 0; i < 10; i++){  
        for(int j = 0; j < 10; j++){ //ITERACION LA CUAL NOS AYUDA A PASAR POR TODOS LOS SOLDADOS DE CADA EJERCITO  
            if(army.get("Soldado" + i + "X" + j) != null){  
                sum += army.get("Soldado" + i + "X" + j).getHealth();  
                count++;  
            }  
        }  
    }  
    if(sum != 0){  
        double avg = sum / (count * 1.0);  
        System.out.println(avg); // DAMOS A CONOCER EL PROMEDIO DE VIDA DE CADA EJERCITO  
        System.out.println("*****");  
        return avg;  
    }else{  
        double avg = 0;  
        System.out.println(avg); // DAMOS A CONOCER EL PROMEDIO DE VIDA DE CADA EJERCITO  
        System.out.println("*****");  
        return avg;  
    }  
}
```

Listing 18: Ejecucion:

```
El Ejercito 1 tiene 2 soldados :  
*****  
Registrando al 1 soldado del Ejercito 1  
-----  
Nombre: Soldado0X1  
Vida: 3  
Fila: 3  
Columna: G  
Registrando al 2 soldado del Ejercito 1  
-----  
Nombre: Soldado1X1  
Vida: 2  
Fila: 9  
Columna: A  
El Ejercito 2 tiene 7 soldados :  
*****  
Registrando al 1 soldado del Ejercito 2  
-----  
Nombre: Soldado0X2
```

```
Vida: 5
Fila: 10
Columna: D
Registrando al 2 soldado del Ejercito 2
-----
Nombre: Soldado1X2
Vida: 4
Fila: 6
Columna: E
Registrando al 3 soldado del Ejercito 2
-----
Nombre: Soldado2X2
Vida: 3
Fila: 2
Columna: H
Registrando al 4 soldado del Ejercito 2
-----
Nombre: Soldado3X2
Vida: 5
Fila: 8
Columna: G
Registrando al 5 soldado del Ejercito 2
-----
Nombre: Soldado4X2
Vida: 1
Fila: 2
Columna: D
Registrando al 6 soldado del Ejercito 2
-----
Nombre: Soldado5X2
Vida: 2
Fila: 2
Columna: C
Registrando al 7 soldado del Ejercito 2
-----
Nombre: Soldado6X2
Vida: 4
Fila: 4
Columna: A

Mostrando tabla de posicion ... --
Leyenda: Ejercito1 --> X | Ejercito2 --> Y
```


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

Listing 19: Ejecucion:

```
*****

El soldado con mayor vida del Ejercito 1 es:

Nombre: Soldado0X1
Vida: 3
Fila: 3
Columna: G
*****

El soldado con mayor vida del Ejercito 2 es:

Nombre: Soldado3X2
Vida: 5
Fila: 8
Columna: G
*****

El promedio de puntos de vida del Ejercito 1 es:
2.5
*****

El promedio de puntos de vida del Ejercito 2 es:
3.4285714285714284
*****
```

4.7. Ejercicio VideoJuego5

- En el septimo commit creamos el metodo rankingBurbujaLife() el cual nos ayuda a ordenar estos soldados de cada ejercito mediante su vida la cual para esto el HashMap su contenido lo pasamos a un array el cual este nos posibilita a que este metodo burubuja sea mas facil de intercambiar para despues imprimir al ejercito con sus respectivos puestos y soldados con su informacion correspondiente para esto iterariamos cada elemento de este HashMap

Listing 20: Commit

```
$ git commit -m "Creamos el metodo rankingBurbujaLife() el cual nos ayuda a ordenar
    estos soldados de cada ejercito mediante su vida la cual para esto el HashMap su
    contenido lo pasamos a un array el cual este nos posibilita a que este metodo
    burubuja sea mas facil de intercambiar para despues imprimir al ejercito con sus
    respectivos puestos y soldados con su informacion correspondiente para esto
    iterariamos cada elemento de este HashMap"
```

Listing 21: Las lineas de codigos del metodo creado:

```
public static void rankingBurbujaLife(HashMap<String, Soldado> army , int num){
    System.out.println("El Ejercito " + num + " ordenando por metodo burbuja: ");
    int count = 0;
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < 10; j++){ //ITERACION LA CUAL NOS AYUDA A PASAR POR TODOS LOS
            SOLDADOS DE CADA EJERCITO
            if(army.get("Soldado" + i + "X" + j) != null){
                count++;
            }
        }
    }
    System.out.println("-----");
    System.out.println("Mostrando Ranking del Ejercito " + num + " ..... ///// --->");
    Soldado[] soldados = new Soldado[count];
    int x = 0;
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < 10; j++){ //ITERACION LA CUAL NOS AYUDA A PASAR POR TODOS LOS
            SOLDADOS AL ARRAY SOLDADO PARA PODER USAR EL USO DEL METODO DE ORDENACION
            BURBUJA
            if(army.get("Soldado" + i + "X" + j) != null){
                if(count - count + x == count){
                    break;
                }else{
                    soldados[count - count + x] = army.get("Soldado" + i + "X" + j);
                }
                x++;
            }
        }
    }
    int n = soldados.length;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1 - i; j++) {
            if (soldados[j].getHealth() < soldados[j + 1].getHealth()) {
                // Intercambiar elementos si estan en el orden incorrecto
                Soldado temp = soldados[j];
                soldados[j] = soldados[j + 1];
                soldados[j + 1] = temp;
            }
        }
    }
    for(int i = 0; i < soldados.length; i++){
        System.out.print("\n" + "Puesto " + (i + 1));
        System.out.println(soldados[i].toString());
        System.out.println("-----");
    }
}
```

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

Listing 22: Ejecucion:

```
El Ejercito 1 tiene 9 soldados :  
*****  
Registrando al 1 soldado del Ejercito 1  
-----  
Nombre: Soldado0X1  
Vida: 1  
Fila: 10  
Columna: D  
Registrando al 2 soldado del Ejercito 1  
-----  
Nombre: Soldado1X1  
Vida: 5  
Fila: 10  
Columna: E  
Registrando al 3 soldado del Ejercito 1  
-----  
Nombre: Soldado2X1  
Vida: 4  
Fila: 10  
Columna: F  
Registrando al 4 soldado del Ejercito 1  
-----  
Nombre: Soldado3X1  
Vida: 4  
Fila: 7  
Columna: B  
Registrando al 5 soldado del Ejercito 1  
-----  
Nombre: Soldado4X1  
Vida: 5  
Fila: 8  
Columna: D  
Registrando al 6 soldado del Ejercito 1  
-----  
Nombre: Soldado5X1  
Vida: 1  
Fila: 3  
Columna: E  
Registrando al 7 soldado del Ejercito 1  
-----  
Nombre: Soldado6X1  
Vida: 1  
Fila: 6  
Columna: G  
Registrando al 8 soldado del Ejercito 1  
-----  
Nombre: Soldado7X1  
Vida: 1  
Fila: 3  
Columna: D  
Registrando al 9 soldado del Ejercito 1
```

```
-----
Nombre: Soldado8X1
Vida: 3
Fila: 10
Columna: G
El Ejercito 2 tiene 7 soldados :
*****
Registrando al 1 soldado del Ejercito 2
-----
Nombre: Soldado0X2
Vida: 1
Fila: 3
Columna: I
Registrando al 2 soldado del Ejercito 2
-----
Nombre: Soldado1X2
Vida: 1
Fila: 6
Columna: I
Registrando al 3 soldado del Ejercito 2
-----
Nombre: Soldado2X2
Vida: 4
Fila: 6
Columna: D
Registrando al 4 soldado del Ejercito 2
-----
Nombre: Soldado3X2
Vida: 1
Fila: 9
Columna: D
Registrando al 5 soldado del Ejercito 2
-----
Nombre: Soldado4X2
Vida: 4
Fila: 9
Columna: I
Registrando al 6 soldado del Ejercito 2
-----
Nombre: Soldado5X2
Vida: 1
Fila: 1
Columna: A
Registrando al 7 soldado del Ejercito 2
-----
Nombre: Soldado6X2
Vida: 3
Fila: 1
Columna: H

Mostrando tabla de posicion ... --
Leyenda: Ejercito1 --> X | Ejercito2 --> Y
```

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

Listing 23: Ejecucion:

```
*****

El soldado con mayor vida del Ejercito 1 es:

Nombre: Soldado4X1
Vida: 5
Fila: 8
Columna: D
*****

El soldado con mayor vida del Ejercito 2 es:

Nombre: Soldado2X2
Vida: 4
Fila: 6
Columna: D
*****

El promedio de puntos de vida del Ejercito 1 es:
2.7777777777777777
*****

El promedio de puntos de vida del Ejercito 2 es:
2.142857142857143
*****

El Ejercito 1 ordenando por metodo burbuja:
-----
Mostrando Ranking del Ejercito 1 ..... //---->

Puesto 1
Nombre: Soldado4X1
Vida: 5
Fila: 8
Columna: D
```

Puesto 2
Nombre: Soldado1X1
Vida: 5
Fila: 10
Columna: E

Puesto 3
Nombre: Soldado3X1
Vida: 4
Fila: 7
Columna: B

Puesto 4
Nombre: Soldado2X1
Vida: 4
Fila: 10
Columna: F

Puesto 5
Nombre: Soldado8X1
Vida: 3
Fila: 10
Columna: G

Puesto 6
Nombre: Soldado7X1
Vida: 1
Fila: 3
Columna: D

Puesto 7
Nombre: Soldado5X1
Vida: 1
Fila: 3
Columna: E

Puesto 8
Nombre: Soldado6X1
Vida: 1
Fila: 6
Columna: G

Puesto 9
Nombre: Soldado0X1
Vida: 1
Fila: 10
Columna: D

```
-----
*****
El Ejercito 2 ordenando por metodo burbuja:
-----
Mostrando Ranking del Ejercito 2 ..... ///// --->

Puesto 1
Nombre: Soldado2X2
Vida: 4
Fila: 6
Columna: D
-----

Puesto 2
Nombre: Soldado4X2
Vida: 4
Fila: 9
Columna: I
-----

Puesto 3
Nombre: Soldado6X2
Vida: 3
Fila: 1
Columna: H
-----

Puesto 4
Nombre: Soldado5X2
Vida: 1
Fila: 1
Columna: A
-----

Puesto 5
Nombre: Soldado0X2
Vida: 1
Fila: 3
Columna: I
-----

Puesto 6
Nombre: Soldado1X2
Vida: 1
Fila: 6
Columna: I
-----

Puesto 7
Nombre: Soldado3X2
Vida: 1
Fila: 9
Columna: D
-----
*****
```

4.8. Ejercicio VideoJuego5

- En el octavo commit creamos el metodo rankingInsercionLife() el cual nos ayuda a ordenar a los soldados pero esta vez de forma inserciva la cual tendria la misma logica que el anterior metodo pero este se aplicaria diferente ya que este tendria su propia funcionalidad la cual la hace unica y despues de esto podemos publicar resultados de cada ejercito el cual va del mayor al menor

Listing 24: Commit

```
$ git commit -m "Creamos el metodo rankingInsercionLife() el cual nos ayuda a ordenar a los soldados pero esta vez de forma inserciva la cual tendria la misma logica que el anterior metodo pero este se aplicaria diferente ya que este tendria su propia funcionalidad la cual la hace unica y despues de esto podemos publicar resultados de cada ejercito el cual va del mayor al menor"
```

Listing 25: Las lineas de codigos del metodo creado:

```
public static void rankingInsercionLife(HashMap<String, Soldado> army , int num){
    System.out.println("El Ejercito " + num + " ordenando por metodo insercion: ");
    int count = 0;
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < 10; j++){ //ITERACION LA CUAL NOS AYUDA A PASAR POR TODOS LOS SOLDADOS DE CADA EJERCITO
            if(army.get("Soldado" + i + "X" + j) != null){
                count++;
            }
        }
    }
    System.out.println("-----");
    System.out.println("Mostrando Ranking del Ejercito " + num + " ..... ///// --->");
    Soldado[] soldados = new Soldado[count];
    int x = 0;
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < 10; j++){ //ITERACION LA CUAL NOS AYUDA A PASAR POR TODOS LOS SOLDADOS AL ARRAY SOLDADO PARA PODER USAR EL USO DEL METODO DE ORDENACION INSERCIÓN
            if(army.get("Soldado" + i + "X" + j) != null){
                if(count - count + x == count){
                    break;
                }else{
                    soldados[count - count + x] = army.get("Soldado" + i + "X" + j); //LA MISMA LOGICA QUE EL ANTERIOR METODO SOLO QUE EN ESTE LO USARIAMOS DE MANERA DIFERENTE YA QUE ESTE SERIA DE FORMA DE INSERCIÓN
                }
                x++;
            }
        }
    }
    int n = soldados.length;
    for (int i = 1; i < n; i++) {
        Soldado actual = soldados[i];
        int j = i - 1;
        while (j >= 0 && soldados[j].getHealth() < actual.getHealth()) { //ORDENAMOS EL EJERCITO RESPECTIVAMENTE MEDIANTE EL METODO QUE NOS OFRECE INSERCIÓN EL CUAL ES ESTE CODIGO
            soldados[j + 1] = soldados[j];
        }
    }
}
```



```
        j--;  
    }  
    soldados[j + 1] = actual;  
}  
for(int i = 0; i < soldados.length; i++){  
    System.out.print("\n" + "Puesto " + (i + 1));  
    System.out.println(soldados[i].toString()); //PUBLICAMOS RESULTADOS  
    System.out.println("-----");  
}  
System.out.println("*****");  
}
```

Listing 26: Ejecucion:

```
*****  
  
El soldado con mayor vida del Ejercito 1 es:  
  
Nombre: Soldado0X1  
Vida: 3  
Fila: 5  
Columna: D  
*****  
  
El soldado con mayor vida del Ejercito 2 es:  
  
Nombre: Soldado0X2  
Vida: 3  
Fila: 4  
Columna: C  
*****  
El promedio de puntos de vida del Ejercito 1 es:  
3.0  
*****  
El promedio de puntos de vida del Ejercito 2 es:  
3.0  
*****  
El Ejercito 1 ordenando por metodo burbuja:  
-----  
Mostrando Ranking del Ejercito 1 ..... /////  
--->  
  
Puesto 1  
Nombre: Soldado0X1  
Vida: 3  
Fila: 5  
Columna: D  
-----  
*****  
El Ejercito 2 ordenando por metodo burbuja:  
-----  
Mostrando Ranking del Ejercito 2 ..... /////  
--->  
  
Puesto 1  
Nombre: Soldado0X2  
Vida: 3  
Fila: 4
```

```

Columna: C
-----
*****
El Ejercito 1 ordenando por metodo insercion:
-----
Mostrando Ranking del Ejercito 1 ..... ///// --->

Puesto 1
Nombre: Soldado0X1
Vida: 3
Fila: 5
Columna: D
-----
*****
El Ejercito 2 ordenando por metodo insercion:
-----
Mostrando Ranking del Ejercito 2 ..... ///// --->

Puesto 1
Nombre: Soldado0X2
Vida: 3
Fila: 4
Columna: C
-----
*****

```

4.9. Estructura de laboratorio 08

- El contenido que se entrega en este laboratorio08 es el siguiente:

```

/Lab08
  "Poner RAMA"

```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

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

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple 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 2: 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

Tabla 3: 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	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 evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	2	
Total		20		18	

6. Referencias

- https://drive.google.com/file/d/1bKx0eCXY6JlkP_RaiD2uaH1JQzoSn6uN/view