Informe de Laboratorio 07

Tema: Laboratorio 07

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
07	Laboratorio 07	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 Laboratorio7
- Usted deberá crear las dos clases Soldado.java y VideoJuego4.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. 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 porejército 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).

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 07.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/VictorMA18/fp2-23b.git`
- URL para el laboratorio 07 en el Repositorio GitHub.
- `https://github.com/VictorMA18/fp2-23b/tree/main/Fase02/Lab07`

4. Actividades del Laboratorio 07

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 VideoJuego4.
- El codigo y el commit seria el siguiente:

Listing 1: Commit

```
$ git commit -m "Publicando la clase soldado para el ejercicio7 la cual es la clase
    soldado donde estan los atributos mas importantes que nos serviran"
```

Listing 2: Las lineas de codigos del metodo creado:

```
// Laboratorio Nro 7 - 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){
       heatlh = p;
    }
    public void setRow(int b){
       row = b;
    }
    public void setColumn(String c){
       column = c;
    }

    // Metodos accesores
    public String getName(){
       return name;
    }
    public int getHealth(){
       return heatlh;

    }
    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 VideoJuego4

- En el segundo commit completamos el metodo arrayfillregister() el cual queremos rellenar el array de soldados del ejercito 2 para poder verlos en el tablero a la ves su informacion de cada soldado el cual va hacer por orden de creacion
- El codigo , el commit y la ejecucion seria el siguiente:

Listing 3: Commit

```
$ git commit -m "Creando el metodo arrayfillregister() para que este pueda imprimir los
    datos de los soldados que estan en el tablero este te devuelve el array lleno con
    la cantidad de soldados y texto que dice su informacion de cada soldado Y tambien
    cumplimos con el cual no se repita un soldado en el mismo casillero ya que al ver
    que este lleno este retrocedera una repeticion ya que estaria tomando la repeticion
    en un casillero lleno"
```

Listing 4: Las lineas de codigos del metodo creado:

```
public static Soldado[][] arrayfillregister(int num){ //METODO CREADO PARA PODER CREAR
    AL EJERCITO 2 EL CUAL USAREMOS LA ESTRUCTURA DE DATO QUE ES EL ARRAY CON TAL QUE
    TAMBIEN REGISTRAMOS
   Random rdm = new Random();
   int numsoldiers = rdm.nextInt(10) + 1;
   System.out.println("La Ejercito " + num + " tiene " + numsoldiers + " soldados:");
   System.out.println("********************************");
   Soldado[][] army = new Soldado[10][10];
   for(int i = 0; i < numsoldiers; i++){ //LOS REGISTRAMOS A CADA UNO POR EL ORDEN DE
       CREACION QUE FUERON CREADOS EL CUAL TAMBIEN COMPLETAMOS SUS DATOS Y LOS
       PUBLICAMOS POR ORDEN
      System.out.println("Registrando al " + (i + 1) + " soldado del Ejercito " + num +
          "");
      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));
      if(army[row - 1][(int)column.charAt(0) - 65] == null){ //VERIFICAMOS QUE NO SE
          REPITAN MISMOS SOLDADOS DE UN EJERCITO EN EL MISMO CUADRADO
         System.out.print("------------------");
         army[row - 1][(int)column.charAt(0) - 65] = new Soldado(name, health, row,
             column);
         System.out.println(army[row - 1][(int)column.charAt(0) - 65].toString());
      }else{
         i -= 1;
      }
   }
   return army;
}
```

Listing 5: Ejecucion:

```
La Ejercito 2 tiene 6 soldados:
********************************
Registrando al 1 soldado del Ejercito 2
------------------
Nombre: Soldado0X2
Vida: 5
Fila: 8
Columna: C
Registrando al 2 soldado del Ejercito 2
------------------
Nombre: Soldado1X2
Vida: 5
Fila: 2
```

```
Columna: I
Registrando al 3 soldado del Ejercito 2
------------------
Nombre: Soldado2X2
Vida: 5
Fila: 5
Columna: F
Registrando al 4 soldado del Ejercito 2
------------------
Nombre: Soldado3X2
Vida: 5
Fila: 2
Columna: G
Registrando al 5 soldado del Ejercito 2
------------------
Nombre: Soldado4X2
Vida: 2
Fila: 10
Columna: G
Registrando al 6 soldado del Ejercito 2
------------------
Nombre: Soldado5X2
Vida: 4
Fila: 9
Columna: F
```

4.3. Ejercicio VideoJuego4

- En el tercer commit creamos el metodo arrayListFillRegister() para que este pueda llenar los ArrayList que creamos para el ejercito 1 en este se creara un arraylist con casillas de soldados con datos nulos el cual se va ir llenando aleatoriamente con soldados y a la vez de esto nos mostrara por orden de creacion la informacion de los soldados a la vez tambien permitiriamos que cada casilla no se repita un mismo soldado ya que este sera verificado mediante si este casilla sea diferente a un soldado nulo
- El codigo , el commit y la ejecucion seria el siguiente:

Listing 6: Commit

```
$ git commit -m "Creamos el metodo arrayListFillRegister() para que este pueda llenar
    los ArrayList que creamos para el ejercito 1 en este se creara un arraylist con
    casillas de soldados con datos nulos el cual se va ir llenando aleatoriamente con
    soldados y a la vez de esto nos mostrara por orden de creacion la informacion de
    los soldados a la vez tambien permitiriamos que cada casilla no se repita un mismo
    soldado ya que este sera verificado mediante si este casilla sea diferente a un
    soldado nulo"
```

Listing 7: Las lineas de codigos del metodo creado:

```
public static ArrayList<ArrayList<Soldado>> arrayListFillRegister(int num){
   Random rdm = new Random();
   ArrayList<ArrayList<Soldado>> army = new ArrayList<ArrayList<Soldado>>();
   int numbersoldiers = rdm.nextInt(10) + 1;
   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 " + numbersoldiers + " soldados :
         " );
     System.out.println("**********************************");
     for(int i = 0; i < numbersoldiers; i++){ //LLENAMOS CASILLAS CON CADA SOLDADO CREADO
         ALEATORIAMENTE
        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 VIDEOJUEGO3.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));
           System.out.println(army.get(row - 1).get((int)column.charAt(0) -
               65).toString());
        }else{
           i -= 1; //NOS AYUDARIA CON LOS SOLDADOS QUE SE REPITEN EN EL MISMO CASILLERO
               CON TAL QUE NO DEBERIA CONTAR
        }
     }
     System.out.println("**********************************");
     return army;
  }
```

Listing 8: Ejecucion:

```
El Ejercito 1 tiene 2 soldados :
**********************************
Registrando al 1 soldado del Ejercito 1
------------------
Nombre: Soldado0X1
Vida: 4
Fila: 2
Columna: F
Registrando al 2 soldado del Ejercito 1
------------------
Nombre: Soldado1X1
Vida: 2
Fila: 8
Columna: I
**********************************

El Ejercito 2 tiene 4 soldados:
**********************************
```

```
Registrando al 1 soldado del Ejercito 2
------------------
Nombre: Soldado0X2
Vida: 1
Fila: 4
Columna: D
Registrando al 2 soldado del Ejercito 2
------------------
Nombre: Soldado1X2
Vida: 4
Fila: 4
Columna: H
Registrando al 3 soldado del Ejercito 2
------------------
Nombre: Soldado2X2
Vida: 2
Fila: 7
Columna: F
Registrando al 4 soldado del Ejercito 2
------------------
Nombre: Soldado3X2
Vida: 2
Fila: 8
Columna: I
```

4.4. Ejercicio VideoJuego4

- En el cuarto commit creamos el metodo viewBoard() el cual nos dejaria permitir visualizar la tabla junto a su leyenda , con los soldados de cada ejercito su posicion la cual del ejercito 1 seria una x y el ejercito 2 seria una y en este tambien aplicamos para cada casilla la cual no sea nula poner una x para el ejercito 1 y en caso contrario debria ser del ejercito 2 el cual seria una y y si tambien seria en caso contrario seria nulo
- El codigo , el commit y la ejecucion seria el siguiente:

Listing 9: Commit

```
$ git commit -m "Creamos el metodo arrayListFillRegister() para que este pueda llenar
    los ArrayList que creamos para el ejercito 1 en este se creara un arraylist con
    casillas de soldados con datos nulos el cual se va ir llenando aleatoriamente con
    soldados y a la vez de esto nos mostrara por orden de creacion la informacion de
    los soldados a la vez tambien permitiriamos que cada casilla no se repita un mismo
    soldado ya que este sera verificado mediante si este casilla sea diferente a un
    soldado nulo"
```

Listing 10: Las lineas de codigos del metodo creado:

```
public static ArrayList<ArrayList<Soldado>> arrayListFillRegister(int num){
   Random rdm = new Random();
   ArrayList<ArrayList<Soldado>> army = new ArrayList<ArrayList<Soldado>>();
   int numbersoldiers = rdm.nextInt(10) + 1;
   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 " + numbersoldiers + " soldados :
            " );
        System.out.println("*********************************");
        for(int i = 0; i < numbersoldiers; i++){ //LLENAMOS CASILLAS CON CADA SOLDADO CREADO
            ALEATORIAMENTE
            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 VIDEOJUEGO3.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));
                System.out.println(army.get(row - 1).get((int)column.charAt(0) -
                    65).toString());
            }else{
                i -= 1; //NOS AYUDARIA CON LOS SOLDADOS QUE SE REPITEN EN EL MISMO CASILLERO
                    CON TAL QUE NO DEBERIA CONTAR
            }
        }
        System.out.println("*********************************");
        return army;
    }
```

Listing 11: Ejecucion:

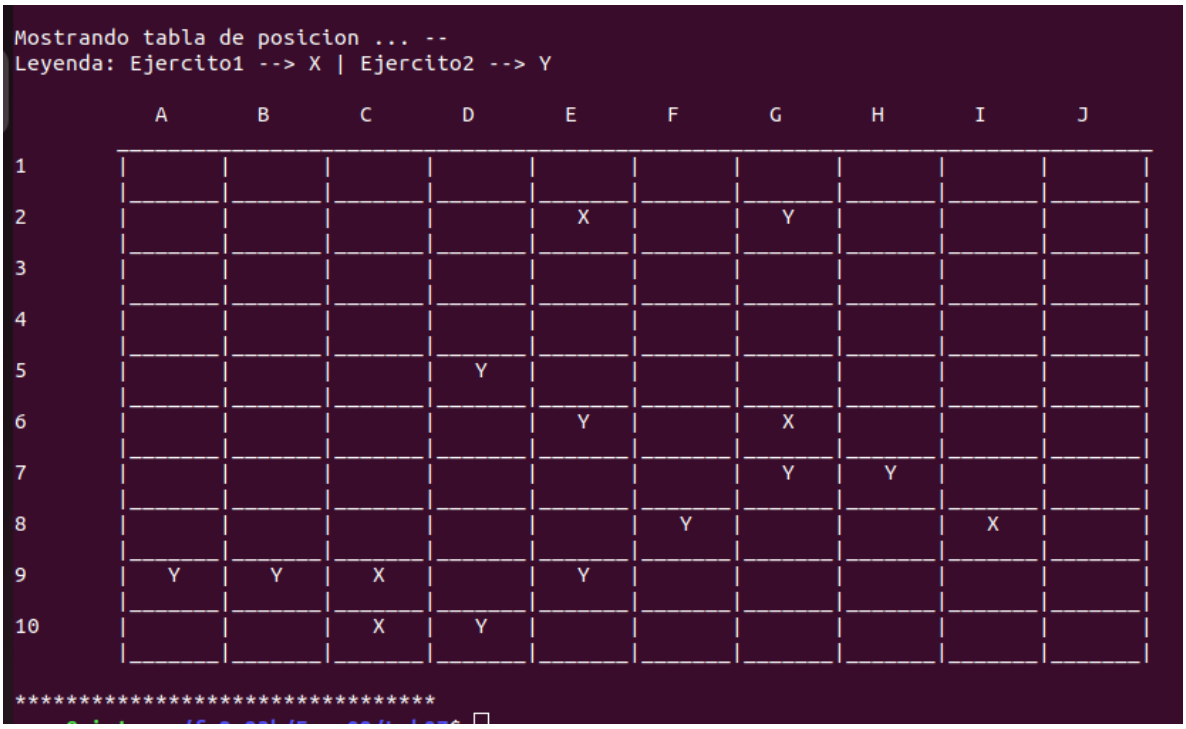
```
El Ejercito 1 tiene 5 soldados :
*********************************
Registrando al 1 soldado del Ejercito 1
------------------
Nombre: Soldado0X1
Vida: 4
Fila: 9
Columna: C
Registrando al 2 soldado del Ejercito 1
------------------
Nombre: Soldado1X1
Vida: 2
Fila: 2
Columna: E
Registrando al 3 soldado del Ejercito 1
------------------
Nombre: Soldado2X1
Vida: 5
Fila: 8
Columna: I
```

```
Registrando al 4 soldado del Ejercito 1
------------------
Nombre: Soldado3X1
Vida: 3
Fila: 6
Columna: G
Registrando al 5 soldado del Ejercito 1
------------------
Nombre: Soldado4X1
Vida: 1
Fila: 10
Columna: C
********************************

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

```
Fila: 5
Columna: D
Registrando al 8 soldado del Ejercito 2
------------------
Nombre: Soldado7X2
Vida: 3
Fila: 9
Columna: B
Registrando al 9 soldado del Ejercito 2
------------------
Nombre: Soldado8X2
Vida: 5
Fila: 7
Columna: G
Registrando al 10 soldado del Ejercito 2
------------------
Nombre: Soldado9X2
Vida: 1
Fila: 9
Columna: E
**********************************

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



4.5. Ejercicio VideoJuego4

- En el quinto commit creamos el metodo arrayListLongerLife() el cual nos permitira conocer del soldado con mayor puntos de vida para esto hacemos una comprobacion con cada uno de estos

para poder despues compararlos gracias a una iteracion sobre todos los soldados de este ejercito y despues de tener el soldado con mayor puntos de vida se imprimira sus datos para ver de quien se trata

- El codigo , el commit y la ejecucion seria el siguiente:

Listing 12: Commit

```
$ git commit -m "Creamos el metodo arrayListLongerLife() el cual nos permitira conocer
    del soldado con mas vida del ejercito 1 para esto hacemos una comprobacion con cada
    uno de estos soldados el cual vamos iterando por cada uno de estos para poder
    despues compararlos y despues de tener el soldado con mayor puntos de vida se
    imprimira sus datos para ver de quien se trata"
```

Listing 13: Las lineas de codigos del metodo creado:

```
public static void arrayListLongerLife(ArrayList<ArrayList<Soldado>> army, int num){
   System.out.println("El soldado con mayor vida del Ejercito " + num + " es: ");
       //METODO CREADO PARA PODER PERMITIRNOS A CONOCER EL SOLDADO CON MAYOR VIDA DE
       CADA EJERCITO
   int mayor = 0;
   Soldado soldier = null;
   for(int i = 0; i < army.size(); i++){
      for(int j = 0; j < army.get(i).size(); j++){
         if(army.get(i).get(j) != null){ //COMPROBACION QUE HACEMOS PARA PODER DECIR QUE
             EL CASILLERO DONDE ESTAMOS ES UN SOLDADO QUE EXISTE
            if(army.get(i).get(j).getHealth() > mayor){ //COMPARAMOS PUNTOS DE VIDA DE
                CADA SOLDADO PARA VER QUIEN ES EL MAYOR
               mayor = army.get(i).get(j).getHealth();
               soldier = army.get(i).get(j);
            }
         }
      }
   }
   System.out.println(soldier.toString());//IMPRIMIMOS SUS DATOS PARA PODER VER DE QUE
       SOLDADO SE TRATA
   System.out.println("**********************************");
}
```

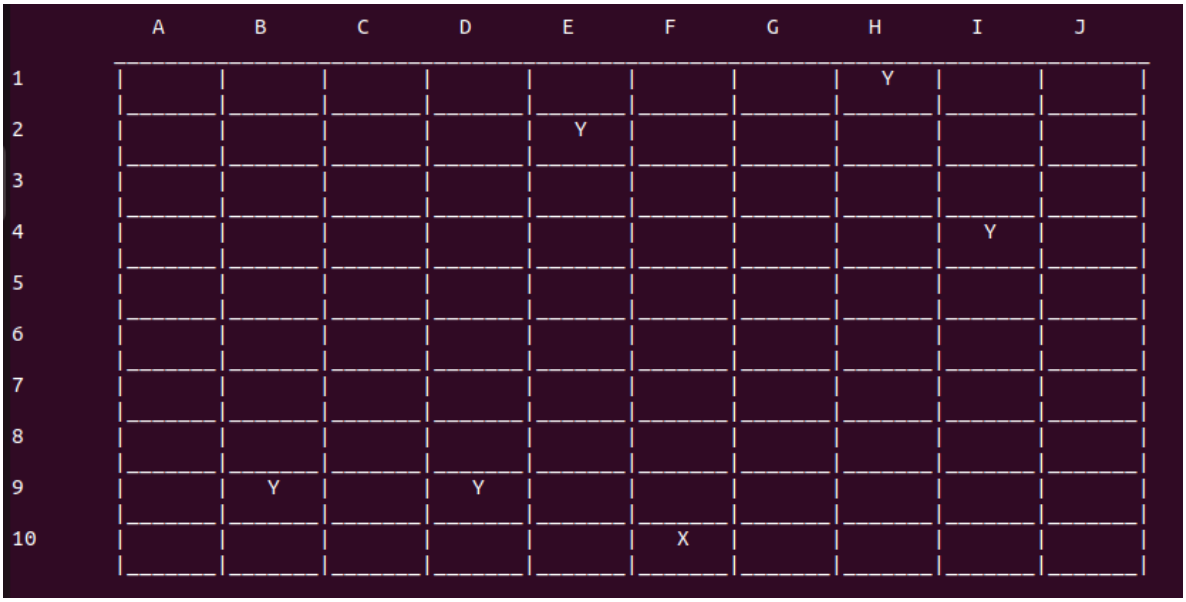
Listing 14: Ejecucion:

```
El Ejercito 1 tiene 1 soldados :
**********************************
Registrando al 1 soldado del Ejercito 1
------------------
Nombre: Soldado0X1
Vida: 1
Fila: 10
Columna: F
**********************************

El Ejercito 2 tiene 5 soldados:
**********************************
Registrando al 1 soldado del Ejercito 2
------------------
```

```
Nombre: Soldado0X2
Vida: 4
Fila: 2
Columna: E
Registrando al 2 soldado del Ejercito 2
------------------
Nombre: Soldado1X2
Vida: 1
Fila: 1
Columna: H
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: 3
Fila: 9
Columna: D
Registrando al 5 soldado del Ejercito 2
------------------
Nombre: Soldado4X2
Vida: 5
Fila: 4
Columna: I
*********************************

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



Listing 15: Ejecucion:

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

Nombre: SoldadoOX1
Vida: 1
Fila: 10
Columna: F
*********************************
```

4.6. Ejercicio VideoJuego4

- En el sexto commit creamos el metodo arrayLongerLife() el cual este nos permite a dar la informacion del soldado con mayor vida del ejercito 2 el cual comparamos con otros soldados que esten en este ejercito y dependiendo de eso va a recoger informacion del soldado con mayor vida y lo guarda en soldier el cual sera imprimido despues.
- El codigo , el commit y la ejecucion seria el siguiente:

Listing 16: Commit

```
$ git commit -m "Creamos el metodo arrayLongerLife() el cual nos imprimira el dato del
    soldado con mayor vida dependiendo de los que esten en el ejercito que va
    comparando con los demas para ver quien es el mayor de todos aplicariamos la misma
    logica que con el metodo arrayListLongerLife() pero en este caso seria para los
    propios array"
```

Listing 17: Las lineas de codigos del metodo creado:

```
public static void arrayLongerLife(Soldado[][] army, int num){
   int mayor = 0;
   Soldado soldier = null; //METODO CREADO QUE NOS VA AYUDAR A DECIRNOS SOBRE LA
       INFORMACION DEL SOLDADO CON MAYOR VIDAD DEL EJERCITO2 EL CUAL TENDREMOS QUE
       ITERAR
   for(int i = 0; i < army.length; i++){
        for(int j = 0; j < army[i].length; j++){//ITERAMOS SOBRE CADA SOLDADO EL CUAL
            COMPARAMOS CON SI ESTE ES EL MAYOR EN CUESTION DE VIDA VAMOS PASANDO POR
            CADA SOLDADO
            if(army[i][j] != null){
                if(mayor < army[i][j].getHealth()){
                    mayor = army[i][j].getHealth();
                    soldier = army[i][j];//ACTUALIZAMOS A ESTE SOLDADO CON EL QUE
                        TIENE MAS VIDA DESPUES LO IMPRIMIMOS 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 18: Ejecucion:

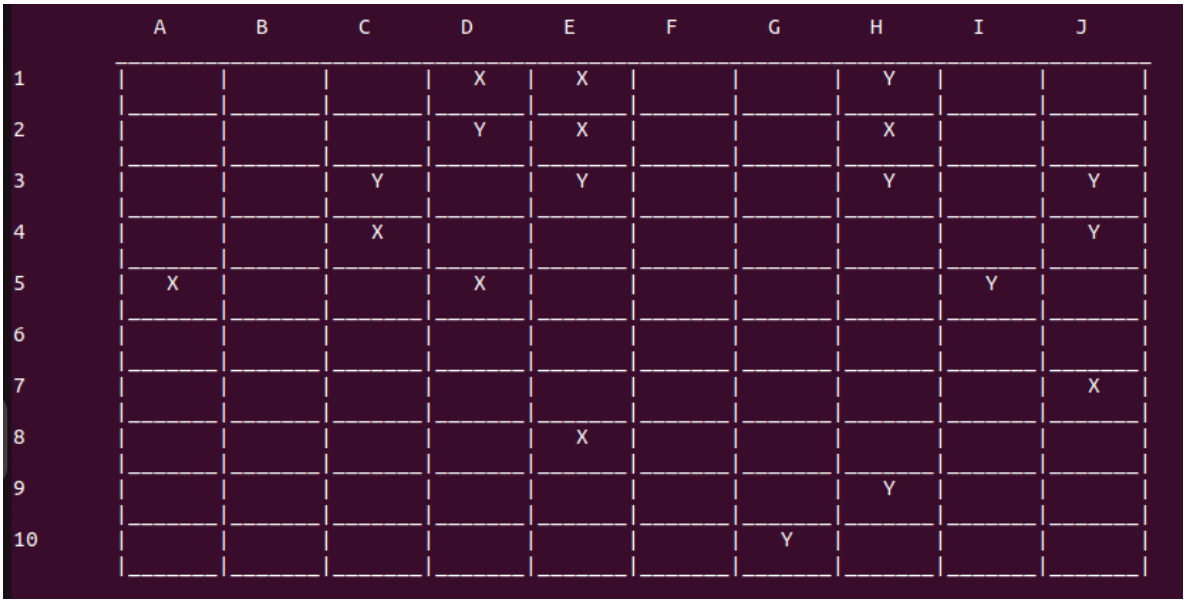
```
El Ejercito 1 tiene 9 soldados :
*********************************
Registrando al 1 soldado del Ejercito 1
------------------
Nombre: Soldado0X1
Vida: 3
Fila: 1
Columna: E
Registrando al 2 soldado del Ejercito 1
------------------
Nombre: Soldado1X1
Vida: 3
Fila: 7
Columna: J
Registrando al 3 soldado del Ejercito 1
------------------
Nombre: Soldado2X1
Vida: 1
Fila: 5
Columna: D
Registrando al 4 soldado del Ejercito 1
------------------
Nombre: Soldado3X1
Vida: 4
Fila: 1
Columna: D
Registrando al 5 soldado del Ejercito 1
------------------
Nombre: Soldado4X1
Vida: 1
Fila: 8
Columna: E
Registrando al 6 soldado del Ejercito 1
------------------
Nombre: Soldado5X1
Vida: 3
Fila: 2
Columna: E
Registrando al 7 soldado del Ejercito 1
------------------
Nombre: Soldado6X1
Vida: 2
Fila: 5
Columna: A
Registrando al 8 soldado del Ejercito 1
------------------
Nombre: Soldado7X1
Vida: 4
Fila: 2
Columna: H
Registrando al 9 soldado del Ejercito 1
------------------
Nombre: Soldado8X1
Vida: 1
Fila: 4
```

```
Columna: C
*********************************

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

```
Vida: 5
Fila: 2
Columna: D
Registrando al 10 soldado del Ejercito 2
------------------
Nombre: Soldado9X2
Vida: 2
Fila: 3
Columna: J
*********************************

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



Listing 19: Ejecucion:

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

Nombre: Soldado3X1
Vida: 4
Fila: 1
Columna: D
*********************************

El soldado con mayor vida del Ejercito 2 es:

Nombre: Soldado8X2
Vida: 5
Fila: 2
Columna: D
*********************************
```

4.7. Estructura de laboratorio 07

- El contenido que se entrega en este laboratorio07 es el siguiente:

```
/Lab07
   "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 cumplio 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 lo items.
- 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 estan 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/128O7v3wmrnl9g0a6BhMTNMnGIL3jTUDz/view