

# Informe de Laboratorio 06

## Tema: Laboratorio 06

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
06	Laboratorio 06	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 09 Octubre 2023	Al 16 Octubre 2023

### 1. Tarea

- Cree un Proyecto llamado Laboratorio6
- Usted deberá crear las dos clases Soldado.java y VideoJuego3.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. Pero ahora el tablero debe ser un ArrayList bidimensional.
- 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 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).

## 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 06.

## 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/VictorMA18/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/VictorMA18/fp2-23b/tree/main/Fase02/Lab06`

## 4. Actividades del Laboratorio 06

### 4.1. Ejercicio Soldado

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

Listing 1: Commit

```
$ git commit -m " Anadimos la clase para poder avanzar con el siguiente ejercicio  
reutilizamos el archivo del laboratorio 5 ya que serian los mismos atributos solo  
que esta vez lo usaremos con un ArrayList bidimensional"
```

Listing 2: Las lineas de codigos del metodo creado:

```
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 accsores
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 VideoJuego3

- En el segundo commit creamos el metodo fillregisterSoldiers() para que este pueda llenar los ArrayList que creamos para cada ejercito 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 3: Commit

```
$ git commit -m "Hacemos el metodo fillregisterSoldiers() para que este pueda llenar  
los ArrayList que creamos para cada ejercito 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 4: Las lineas de codigos del metodo creado:

```
// Laboratorio Nro 6 - Ejercicio Videojuego3  
// Autor: Mamani Anahua Victor Narciso  
// Colaboro:  
// Tiempo:  
import java.util.*;  
class VideoJuego3{  
    public static ArrayList<ArrayList<Soldado>> fillregisterSoldiers(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  
            System.out.print("Registrando al " + (i + 1) + " soldado del Ejercito " + num +  
            "");  
            String name = "Soldado" + i + "X" + num;  
            //System.out.println(name); PRUEBA QUE SE HIZO PARA VER LOS NOMBRES  
            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 VIDEOJUEGO2.JAVA YA QUE TENDRIAN LA MISMA  
            FUNCIONALIDAD  
            if(army.get(row - 1).get((int)column.charAt(0) - 65) == null){  
                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());  
                System.out.println("-----");  
            }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;
    }
    public static void main(String args[]){
        ArrayList<ArrayList<Soldado>> army1 = fillregisterSoldiers(1);
        ArrayList<ArrayList<Soldado>> army2 = fillregisterSoldiers(2);
    }
}
```

Listing 5: La ejecución:

```
El Ejercito 1 tiene 3 soldados :

Registrando al 1 soldado del Ejercito 1
Nombre: Soldado0X1
Vida: 4
Fila: 10
Columna: H
-----
Registrando al 2 soldado del Ejercito 1
Nombre: Soldado1X1
Vida: 1
Fila: 5
Columna: I
-----
Registrando al 3 soldado del Ejercito 1
Nombre: Soldado2X1
Vida: 5
Fila: 5
Columna: G
-----
*****
El Ejercito 2 tiene 6 soldados :

Registrando al 1 soldado del Ejercito 2
Nombre: Soldado0X2
Vida: 3
Fila: 1
Columna: F
-----
Registrando al 2 soldado del Ejercito 2
Nombre: Soldado1X2
Vida: 5
Fila: 3
Columna: G
-----
Registrando al 3 soldado del Ejercito 2
Nombre: Soldado2X2
Vida: 5
Fila: 6
Columna: D
-----
Registrando al 4 soldado del Ejercito 2
Nombre: Soldado3X2
Vida: 4
Fila: 4
```

```
Columna: I
-----
Registrando al 5 soldado del Ejercito 2
Nombre: Soldado4X2
Vida: 1
Fila: 9
Columna: B
-----
Registrando al 6 soldado del Ejercito 2
Nombre: Soldado5X2
Vida: 5
Fila: 3
Columna: J
-----
*****
```

### 4.3. Ejercicio VideoJuego3

- En el tercer commit realizamos 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 ejecución seria el siguiente:

Listing 6: Commit

```
$ git commit -m "Creamos el metodo viewBoard() el cual nos dejaria permitir visualizar la tabla 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"
```

Listing 7: Las lineas de codigos del metodo creado:

```
public static void viewBoard(ArrayList<ArrayList<Soldado>> army1,
    ArrayList<ArrayList<Soldado>> army2){ //EN ESTE METODO NOS AYUDAMOS DEL ANTERIOR
    ARCHIVO YA QUE TENDRIAMOS UNA BASE PARA PODER CREARLO
    System.out.println("Mostrando 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 < army1.size(); 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 < army1.get(i).size(); j++){
            if(army1.get(i).get(j) != null){
                System.out.print("| " + "X" + " "); //VERIFICANDOLA POSICIONES DE CADA
                SOLDADO DE CADA EJERCITO CON SU RESPECTIVO INDICADOR PARA PODER
                UBICARLOS
            } else {
                System.out.print("| " + "Y" + " ");
            }
        }
        System.out.println();
    }
}
```

```
        }else if(army2.get(i).get(j) != null){
            System.out.print("| " + "Y" + " ");
        }else{
            System.out.print("| ");
        }
    }
    System.out.println("|");
    System.out.println("\t|-----|-----|-----|-----|-----|-----|-----|-----|");
}
System.out.println("\n*****");
}
```

Listing 8: La ejecución: **PARA PODER VER LA TABLA COMPLETAMENTE MIRAR LA PLANTILLA DE LATEX O EJECUTARLO:**

```
El Ejercito 1 tiene 2 soldados :

Registrando al 1 soldado del Ejercito 1
Nombre: Soldado0X1
Vida: 2
Fila: 4
Columna: A
-----

Registrando al 2 soldado del Ejercito 1
Nombre: Soldado1X1
Vida: 1
Fila: 2
Columna: D
-----

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

Registrando al 1 soldado del Ejercito 2
Nombre: Soldado0X2
Vida: 5
Fila: 10
Columna: A
-----

Registrando al 2 soldado del Ejercito 2
Nombre: Soldado1X2
Vida: 2
Fila: 10
Columna: H
-----

Registrando al 3 soldado del Ejercito 2Registrando al 3 soldado del Ejercito 2
Nombre: Soldado2X2
Vida: 1
Fila: 4
Columna: B
-----

Registrando al 4 soldado del Ejercito 2
Nombre: Soldado3X2
Vida: 2
Fila: 6
Columna: F
-----
```

\*\*\*\*\*

Mostrando tabla de posicion ... --

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

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

\*\*\*\*\*

#### 4.4. Estructura de laboratorio 06

#### 4.5. Ejercicio VideoJuego3

- En el cuarto commit realizamos el metodo longerLife() el cual nos permitira conocer del soldado de cada ejercito con mayor puntos de vida para esto hacemos una comprobacion con cada uno de estos para poder despues compararlos eny despues de tener el soldado con mayor puntos de vida se imprimira sus datos para ver de quien se trata y a que ejercito
- El codigo , el commit y la ejecucion seria el siguiente:

Listing 9: Commit

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

Listing 10: Las lineas de codigos del metodo creado:

```
public static void longerLife(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()); //IMPRIMOS SUS DATOS PARA PODER VER DE QUE
SOLDADO SE TRATA
System.out.println("*****");
}
```

Listing 11: La ejecución: PARA PODER VER LA TABLA COMPLETAMENTE MIRAR LA PLAN-  
TILLA DE LATEX O EJECUTARLO:

```
El Ejercito 1 tiene 4 soldados :

Registrando al 1 soldado del Ejercito 1
Nombre: Soldado0X1
Vida: 5
Fila: 1
Columna: I
-----
Registrando al 2 soldado del Ejercito 1
Nombre: Soldado1X1
Vida: 1
Fila: 7
Columna: A
-----
Registrando al 3 soldado del Ejercito 1
Nombre: Soldado2X1
Vida: 4
Fila: 1
Columna: E
-----
Registrando al 4 soldado del Ejercito 1
Nombre: Soldado3X1
Vida: 5
Fila: 6
Columna: H
-----
*****
El Ejercito 2 tiene 4 soldados :

Registrando al 1 soldado del Ejercito 2
Nombre: Soldado0X2
Vida: 4
Fila: 5
Columna: A
```

-----  
Registrando al 2 soldado del Ejercito 2

Nombre: Soldado1X2

Vida: 2

Fila: 5

Columna: G

-----  
Registrando al 3 soldado del Ejercito 2

Nombre: Soldado2X2

Vida: 3

Fila: 1

Columna: A

-----  
Registrando al 4 soldado del Ejercito 2

Nombre: Soldado3X2

Vida: 3

Fila: 10

Columna: J

-----  
\*\*\*\*\*

Mostrando tabla de posicion ... --

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

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

\*\*\*\*\*

El soldado con mayor vida del Ejercito 1 es:

Nombre: Soldado0X1

Vida: 5

Fila: 1

Columna: I

\*\*\*\*\*

El soldado con mayor vida del Ejercito 2 es:

Nombre: Soldado0X2  
Vida: 4  
Fila: 5  
Columna: A  
\*\*\*\*\*

#### 4.6. Estructura de laboratorio 06

- El contenido que se entrega en este laboratorio06 es el siguiente:

/Lab06      "MOSTRAR RAMA"

### 5. Rúbricas

#### 5.1. Entregable Informe

Tabla 1: Tipo de Informe

<b>Informe</b>	
<b>Latex</b>	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	1	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
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	
<b>Total</b>		20		14	

## 6. Referencias

- <https://drive.google.com/file/d/18wvjXuguiRaIZ3Z0dElzC-LM9hrabhue/view>