
	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programacion 2				
TÍTULO DE LA PRÁCTICA:	Laboratorio 10				
NÚMERO DE PRÁCTICA:	10	AÑO LECTIVO:	1	NRO. SEMESTRE:	2
FECHA DE PRESENTACIÓN	20/12/2023	HORA DE PRESENTACIÓN	4:00 pm		
INTEGRANTE (s): Cuno Cahuari Armando Steven				NOTA:	
DOCENTE(s): Alain Revilla Christian					



SOLUCIÓN Y RESULTADOS	
<b>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</b>	
<p>Para este laboratorio se requirió lo mismo del laboratorio anterior pero que en esta oportunidad se simule un juego entre 2 personas para ellos hemos necesitado diversos métodos los cuales nos ayuden tanto para ubicar el movimiento y la batalla que puede darse si se encuentra con un enemigo para ello</p> <p>Empezamos con el metodo Juego:</p>	

```
public static void juego(ArrayList<Soldado> ejercito1, ArrayList<Soldado> ejercito2, char[][]tablero){
    boolean validez = true;
    String coordenada;
    int movimiento, fila, columna;
    int i = 0;
    while(validez){
        if(ejercito1.size() == 0 || ejercito2.size() == 0){
            validez = false;
            break;
        } else if(i % 2 == 0){
            System.out.println("Turno del Ejercito 1");
            coordenada = ingresar(ejercito1.get(0).getFigura(), tablero);
            movimiento = ingresarMovimiento(coordenada);
            movimientoJugado(coordenada, movimiento, ejercito1, ejercito2, tablero);
            impTablero(tablero);
        } else {
            System.out.println("Turno del Ejercito 2");
            coordenada = ingresar(ejercito2.get(0).getFigura(), tablero);
            movimiento = ingresarMovimiento(coordenada);
            movimientoJugado(coordenada, movimiento, ejercito2, ejercito1, tablero);
            impTablero(tablero);
        }
        i++;
    }
    if(ejercito1.size() == 0){
        System.out.println("Salio victorio el ejercito 2");
        for(Soldado n: ejercito2){
            System.out.println(n);
        }
    } else{
        System.out.println("Salio victorioso el ejercito 1");
        for(Soldado m: ejercito2){
            System.out.println(m);
        }
    }
}
```

Como vemos en el código que posee el método juego se ve el uso de diversos métodos necesarios para que se realice un juego entre un humano versus otra asimismo para ir intercambiando los turnos fue necesario mediante la variable i que va intercambiando cada jugador. Asimismo, el 1er if que hay dentro del while se encarga de ver si los ArrayList son de tamaño 0 esto nos indicara que un jugador gano la partida rompiendo el ciclo while mediante el cambio de valor de validez a false.

Empecemos con los turnos para ello:

```
} else if(i % 2 == 0){
    System.out.println("Turno del Ejercito 1");
    coordenada = ingresar(ejercito1.get(0).getFigura(), tablero);
    movimiento = ingresarMovimiento(coordenada);
    movimientoJugado(coordenada, movimiento, ejercito1, ejercito2, tablero);
    impTablero(tablero);
} else {
    System.out.println("Turno del Ejercito 2");
    coordenada = ingresar(ejercito2.get(0).getFigura(), tablero);
    movimiento = ingresarMovimiento(coordenada);
    movimientoJugado(coordenada, movimiento, ejercito2, ejercito1, tablero);
    impTablero(tablero);
}
i++;
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

Como se redactó anteriormente la variable *i* se encarga de ir turnando a ambos ArrayList de Soldado asimismo para que este juego sea didáctico fue necesario el apoyo de métodos los cuales simulan un juego:



El método `ingresar()`:

```
public static String ingresar(char n, char[][] tablero){
    String coordenada = "";
    int fila, columna;
    boolean validez = true;
    Scanner sc = new Scanner(System.in);
    while(validez){
        System.out.println("Ingrese la coordenada");
        coordenada = sc.next();
        fila = Integer.parseInt(coordenada.substring(1, coordenada.length()));
        columna = nroColumna(coordenada.charAt(0));
        System.out.println(fila + " " + columna);
        if(tablero[fila - 1][columna] == n){
            validez = false;
        } else {
            System.out.println("Coordenada incorrecta");
        }
    }
    return coordenada;
}
```

Este tiene el rol de verificar que la coordenada ingresada como "A5" "B9" entre otros mediante un String tenga el valor correspondiente al turno del ejercito con el cual se esta empezando el turno. Si en caso el valor no es el correspondiente el while preguntara constantemente hasta que este llegue a colocar la coordenada correspondiente. Aca asimismo se utilizó el método `nroColumna` que solo transforma el índice 0 de la coordenada en un numero para ubicarlo en el arreglo bidimensional de char.

```
public static int nroColumna(char n){
    switch(n){
        case 'A': return 0;
        case 'B': return 1;
        case 'C': return 2;
        case 'D': return 3;
        case 'E': return 4;
        case 'F': return 5;
        case 'G': return 6;
        case 'H': return 7;
        case 'I': return 8;
        case 'J': return 9;
        default: return 0;
    }
}
```

El método `ingresarMovimiento()`:

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

```

public static int ingresarMovimiento(String coordenada){
    int movimiento = 0;
    boolean validez = true;
    int fila, columna;
    Scanner sc = new Scanner(System.in);
    while(validez){
        System.out.println("1 2 3\n8 x 4\n7 6 5");
        System.out.println("Ingresar Movimiento: ");
        movimiento = sc.nextInt();
        fila = conversionFila(coordenada.substring(1, coordenada.length())) - 1;
        columna = nroColumna(coordenada.charAt(0));
        fila = Movimiento.movFila(fila, movimiento);
        columna = Movimiento.movColumna(columna, movimiento);
        System.out.println(fila + " Movimiento aplicado " + columna);
        if(validezMovimiento(fila, columna)){
            validez = false;
        } else {
            System.out.println("Movimiento invalido");
            fila = Movimiento.restFila(fila, movimiento);
            columna = Movimiento.restColumna(columna, movimiento);
        }
    }
    return movimiento;
}

```

Tiene la misma función que el método anterior solo que este mediante la coordenada que recibe se ubicara en 2 variables tanto como en la posición fila y la posición columna. Con ayuda de métodos de clase para que no sea tan confuso el código hacemos simulaciones de que el movimiento dado este en el rango requerido para ubicarse en el tablero esto es gracias al método validezMovimiento:

```



public static boolean validezMovimiento(int fila, int columna){
    return fila <= 9 && fila >= 0 && columna <= 9 && columna >= 0;
}

```

Asimismo, el while que tiene el método ingresarMovimiento seguirá en ciclo hasta que ingresemos el movimiento correcto asimismo en la parte del else al ver que es un movimiento invalido regresamos a los valores originales mediante los mismos métodos de clase.

```
public class Movimiento {  
    public static int movFila(int fila, int mov){  
        switch(mov){  
            case 1: return fila -= 1;  
            case 2: return fila -= 1;  
            case 3: return fila -= 1;  
            case 5: return fila += 1;  
            case 6: return fila += 1;  
            case 7: return fila += 1;  
            default: return fila;  
        }  
    }  
    public static int restFila(int fila, int mov){  
        switch(mov){  
            case 1: return fila += 1;  
            case 2: return fila += 1;  
            case 3: return fila += 1;  
            case 5: return fila -= 1;  
            case 6: return fila -= 1;  
            case 7: return fila -= 1;  
            default: return fila;  
        }  
    }  
    public static int movColumna(int columna, int mov){  
        switch(mov){  
            case 3: return columna += 1;  
            case 4: return columna += 1;  
            case 5: return columna += 1;  
            case 1: return columna -= 1;  
            case 7: return columna -= 1;  
            case 8: return columna -= 1;  
            default: return columna;  
        }  
    }  
    public static int restColumna(int columna, int mov){  
        switch(mov){  
            case 3: return columna -= 1;  
            case 4: return columna -= 1;  
            case 5: return columna -= 1;  
            case 1: return columna += 1;  
            case 7: return columna += 1;  
            case 8: return columna += 1;  
            default: return columna;  
        }  
    }  
}
```

Ahora pasamos al método movimientoJugado para este método ya haremos una simulación de que el movimiento ya se realiza asimismo se reconocerá que la posiciona a donde se movilizara es válida o si hay una pieza amiga, entre otros.

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 6</p>

```

public static void movimientoJugado(String coordenada, int movimiento, ArrayList<Soldado> usuario, ArrayList<Soldado>
int fila_act, columna_act, fila_mov, columna_mov, posAct;
fila_act = conversionFila(coordenada.substring(1, coordenada.length())) - 1;
columna_act = nroColumna(coordenada.charAt(0));
fila_mov = Movimiento.movFila(fila_act, movimiento);
columna_mov = Movimiento.movColumna(columna_act, movimiento);
if(tablero[fila_mov][columna_mov] == usuario.get(0).getFigura()){
    System.out.println("Posicion ocupada por una figura de tu mismo ejercito");
} else if(tablero[fila_mov][columna_mov] == '-'){
    posAct = buscadorPosicion(usuario, coordenada);
    usuario.get(posAct).setFila(fila_mov + 1);
    usuario.get(posAct).setCol(conversionBusqueda(columna_mov));
    tablero[fila_act][columna_act] = '-';
    tablero[fila_mov][columna_mov] = usuario.get(0).getFigura();
} else {
    System.out.println("Enemigo hallado");
    batalla(coordenada, movimiento, usuario, contrincante, tablero);
}
}

```

Mediante las variables fila\_act , columna\_act, fila\_mov, columna\_mov haremos una simulación de que probabilidades la pieza del destino tendrá asimismo tiende a tener 3 posibilidades uno cuando se encuentra con '-' pasa a ocupar ese espacio, otra cuando se encuentra con una figura aliada por lo cual no es valido y la tercera cuando se encuentra un enemigo y se da a realizar una batalla. En cada movimiento que va a realizar el Soldado es necesario ir actualizando sus variables de encapsulamiento posFila y posColumna para ellos lo realizamos mediante sets. Para poder establecer estos sets es necesario saber que índice del ArrayList es la que va a ejecutar ese movimiento mediante un método buscadorPosicion nos encontrara cual es el índice que esta ubicado.

```

public static int buscadorPosicion(ArrayList<Soldado> ejercito, String posicion){
    for(int i = 0; i < ejercito.size(); i++){
        if(posicion.charAt(0) == ejercito.get(i).getColumna() && conversionFila(posicion.substring(1, posicion.length())) == i){
            return i;
        }
    }
    return 0;
}

```

Como se realizaron tanto en simulación con amigo y en un espacio vacio entonces ahora vamos al método batalla la cual es necesario para hallar al vencedor:



```
public static void batalla(String coordenada, int movimiento, ArrayList<Soldado> usuario, ArrayList<Soldado> contr
    int fila_act, columna_act, fila_mov, columna_mov, posAct, us, cont;
    fila_act = conversionFila(coordenada.substring(1, coordenada.length())) - 1;
    columna_act = nroColumna(coordenada.charAt(0));
    fila_mov = Movimiento.movFila(fila_act, movimiento);
    columna_mov = Movimiento.movColumna(columna_act, movimiento);
    String movRealizado = conversionBusqueda(columna_mov) + String.valueOf(fila_mov + 1);
    us = buscadorPosicion(usuario, coordenada);
    usuario.get(us).atacar();
    cont = buscadorPosicion(contrincante, movRealizado);
    contrincante.get(cont).defender();
    System.out.println("Jugador actual : " + usuario.get(us).getNivVidAct());
    System.out.println("Jugador enemigo : " + contrincante.get(cont).getNivVidAct());
    if(usuario.get(us).getNivVidAct() == contrincante.get(cont).getNivVidAct()){
        int suerte = (int)(Math.random() * 2);
        if(suerte == 0){
            contrincante.remove(cont);
            usuario.get(us).setFila(fila_mov + 1);
            usuario.get(us).setCol(conversionBusqueda(columna_mov));
            tablero[fila_mov][columna_mov] = usuario.get(us).getFigura();
            tablero[fila_act][columna_act] = '-';
        } else {
            usuario.remove(us);
            tablero[fila_act][columna_act] = '-';
        }
    }
    } else if(usuario.get(us).getNivVidAct() > contrincante.get(cont).getNivVidAct()){
        contrincante.remove(cont);
        usuario.get(us).setFila(fila_mov + 1);
        usuario.get(us).setCol(conversionBusqueda(columna_mov));
        tablero[fila_mov][columna_mov] = usuario.get(us).getFigura();
        tablero[fila_act][columna_act] = '-';
    } else {
        usuario.remove(us);
        tablero[fila_act][columna_act] = '-';
    }
}
```

Como ya tenemos la coordenada de la ubicación del jugador actual asimismo el movimiento valido que realizara entonces al encontrarse con un enemigo haremos una búsqueda de índice de ambos valores mediante el método buscadorPosición necesario para rescatar el índice de estos ambos soldados para mediante el método de instancia get hallar el nivel de vida y dar lugar a la batalla. Teniendo en cuenta que si ambos jugadores tienen el mismo valor de vida se decide a la suerte mediante un Math.random pero en otros casos si uno tiene mayor vida al otro este eliminando del arraylist el método conversionBusqueda se encarga de hallar la posición del int regresando un char necesario para que este sea válido.



Ejemplo:

<div><div>Inicio de Juego El tablero es : -----* --*--**-- -*-----&amp; -----*-- ----- --*----- --*----- ----- -&amp;----- *----- Turno del Ejercito 1 Ingrese la coordenada D2</div></div>	
---	--



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

<pre> El tablero es : -----*-----* -----**----- -*-----&amp; -----*----- ----- --*----- --*----- ----- -&amp;----- *----- Turno del Ejercito 2 Ingrese la coordenada J3 3      9 1 2 3 8 x 4 7 6 5 Ingresar Movimiento: 1 1 Movimiento aplicado      8 El tablero es : -----*-----* -----**&amp;----- -*----- -----*----- ----- --*----- --*----- ----- -&amp;----- *----- Turno del Ejercito 1 Ingrese la coordenada H2 2      7 1 2 3 </pre>	
--	--

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

```

8 x 4
7 6 5
Ingresar Movimiento:
4
1 Movimiento aplicado      8
Enemigo hallado
Jugador actual : 2
Jugador enemigo : 5
El tablero es :
---*---*
---*-&---
-*-----
---*---
-----
--*-----
--*-----
-----
-&-----
*-----
Turno del Ejercito 2

```

## II. CONCLUSIONES

El uso de diversos métodos asimismo de otra clase que nos aporta con métodos de clase es bastante bueno para no equivocarnos con diversos métodos, asimismo la lógica del programa debe ser mas entendible.

## RETROALIMENTACIÓN GENERAL

Es necesario analizar y leer la documentación que nos ofrece Java con respecto a la programación ya que nos ofrece diversidad de métodos que ya estén creados para que nosotros no estemos creando otros por encima asimismo mejorar en los nombre de identificadores que tienen las variables en los diversos métodos para que no sea confuso.

## REFERENCIAS Y BIBLIOGRAFÍA