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

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	FUNDAMENTOS DE PROGRAMACIÓN 2				
TÍTULO DE LA PRÁCTICA:	Arreglos Bidimensionales de Objetos				
NÚMERO DE PRÁCTICA:	5	AÑO LECTIVO:	2024	NRO. SEMESTRE:	2024-B
FECHA DE PRESENTACIÓN	18/10/2024	HORA DE PRESENTACIÓN	18/00/00		
INTEGRANTE (s) Riveros Vilca Alberth Edwar				NOTA (0-20)	
DOCENTE(s): Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <ol style="list-style-type: none"> 1. Cree un Proyecto llamado Laboratorio5 2. Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo desarrollado en Laboratorio 3 y 4. 3. Del Soldado nos importa el nombre, nivel de vida, fila y columna (posición en el tablero). 4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el tablero debe ser un arreglo bidimensional de objetos. 5. Inicializar el tablero con n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: Soldado0, Soldado1, etc., un valor de nivel de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (verificar que no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (usar caracteres como _ y otros). Además, mostrar los datos del Soldado con mayor nivel de vida, el promedio de nivel de vida de todos los soldados creados, el nivel de vida de todo el ejército, los datos de todos los soldados en el orden que fueron creados y un ranking de poder de todos los soldados creados, del que tiene más nivel de vida al que tiene menos (usar al menos 2 algoritmos de ordenamiento).

```
public class Soldado { 30 usages new *
    private String nombre; 4 usages
    private int fila; 4 usages
    private int columna; 4 usages
    private int nivelVida; 4 usages
    // Metodos mutadores
    public Soldado(String nombre, int fila, int columna, int nivelVida) { 1 usage
        this.nombre = nombre;
        this.fila = fila;
        this.columna = columna;
        this.nivelVida = nivelVida;
    }
    public void setNombre(String n){ no usages new *
        nombre = n;
    }
    public void setFila(int f){ no usages new *
        fila = f;
    }
    public void setColumna(int c){ no usages new *
        columna = c;
    }

    public void setNivelVida(int p){ no usages new *
        nivelVida = p;
    }
    // Metodos accesoros
    public String getNombre(){ 2 usages new *
        return nombre;
    }
    public int getFila(){ no usages new *
        return fila;
    }
    public int getColumna(){ no usages new *
        return columna;
    }

    public int getNivelVida(){ 7 usages new *
        return nivelVida;
    }
    @Override new *
    public String toString(){
        return "[Nombre: "+nombre+"\tFila: "+(fila+1)+"\tColumna: "+(columna+1)+
            "\tnivel de Vida: "+ nivelVida +"]"+"\\n";
    }
}
```



UNIVERSIDAD NACIONAL DE SAN AGUSTIN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 3

--

```
import java.util.*;

public class VideoJuego2 { new *
    public static void main(String[] args) { new *
        Random rand = new Random();
        boolean[][] casillasOcupadas = new boolean[10][10];
        Soldado[][] soldados = new Soldado[10][10];
        int numSoldados = rand.nextInt( bound: 10) + 1;
        String[] ordenSoldados = new String[numSoldados];
        int count=0;
        do{
            int randColumn = rand.nextInt( bound: 10);
            int randRow = rand.nextInt( bound: 10);
            /*Creacion y verificacion de posicion de los soldados*/
            while(isFull(randRow,randColumn,casillasOcupadas)){
                randColumn = rand.nextInt( bound: 10);
                randRow = rand.nextInt( bound: 10);
            }
            casillasOcupadas[randRow][randColumn] = true;
            soldados[randRow][randColumn] = new Soldado( nombre: "Soldado"+count,randRow,randColumn,
                                                         nivelVida: rand.nextInt( bound: 5)+1);
            ordenSoldados[count] = soldados[randRow][randColumn].getNombre();
            count++;
        }while(count<numSoldados);

        showBoard(casillasOcupadas);

        String mayorVida = findMaxLifeSoldier(soldados);
        System.out.print("Soldado de Mayor Vida: "+mayorVida);

        double promedioVida = calcularPromedioVida(soldados, numSoldados);
        int totalVida = calcularTotalVida(soldados);
        System.out.println("Promedio de nivel de vida: "+promedioVida);
        System.out.println("Total de vida del ejercito: "+totalVida);

        System.out.println("Soldados en el orden de creación");
        armyCreation(ordenSoldados,soldados);

        Soldado[] soldadosFila = toUnidimensional(soldados);
        bubbleSortLife(soldadosFila);
        System.out.println("Ranking de poder (Bubble Sort):");
        showArmyInfo(soldadosFila);

        insertionSortLife(soldadosFila);
        System.out.println("Ranking de poder (Insertion Sort):");
        showArmyInfo(soldadosFila);
    }
}
```

```
/*Verifica si la posicion esta ocupada*/
public static boolean isFull(int row, int column, boolean[][] casillasOcupadas) {
    return casillasOcupadas[row][column];
}

/*Metodo para imprimir la tabla*/
public static void showBoard(boolean[][] casillasOcupadas) { 1usage new *
    System.out.print("\t");
    for(char i = 'A'; i<'K'; i++){
        System.out.print(i+" ");
    }
    System.out.println();
    System.out.print(" ");
    for(int l = 0; l<12; l++){
        System.out.print("----");
    }
    System.out.println();
    for(int j = 0; j<10; j++){
        if(j!=9)
            System.out.print((j+1)+" ");
        else
            System.out.print((j+1)+" ");
        for(int k = 0; k<10; k++){
            System.out.print("|");
            if(isFull(j,k,casillasOcupadas)){
                System.out.print("+");
            }else{
                System.out.print(" ");
            }
            System.out.print("| ");
        }
        System.out.println();
        System.out.print(" ");
        for(int l = 0; l<12; l++){
            System.out.print("----");
        }
        System.out.println();
    }
}

/*Metodo para mostrar los soldados en su orden de creacion*/
public static void armyCreation(String[] orden, Soldado[][] ejercito) { 1usage ne
    for (String nombre : orden) {
        Soldado soldado = findSoldier(nombre, ejercito);
        /*Verifica que no sea nulo*/
        if (soldado != null) {
            System.out.print(soldado.toString());
        }
    }
}
```

```
public static Soldado findSoldier(String nombre, Soldado[][] ejercito) { 1 usage ne
    for (Soldado[] fila : ejercito) {
        for (Soldado soldado : fila) {
            /*Verifica que no sea nulo*/
            if (soldado != null && nombre.equals(soldado.getNombre())) {
                return soldado;
            }
        }
    }
    return null;
}

/*Metodo para mostrar la informacion del ejercito*/
public static void showArmyInfo(Soldado[] ejercito) { 2 usages new *
    for (Soldado soldado : ejercito) {
        if (soldado != null) { /*Verifica que no sea nulo*/
            System.out.print(soldado);
        }
    }
}

/*Metodo para mostrar la vida total del ejercito*/
public static int calcularTotalVida(Soldado[][] soldados) { 2 usages new *
    int totalVida = 0;
    for (Soldado[] fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null) { /*Verifica que no sea nulo*/
                totalVida += soldado.getNivelVida();
            }
        }
    }
    return totalVida;
}

/*Metodo para mostrar la vida promedio del soldado*/
public static double calcularPromedioVida(Soldado[][] soldados, int numSoldados) {
    int totalVida = calcularTotalVida(soldados);
    return (double) totalVida / numSoldados;
}

/*Metodo para mostrar el soldado de mayor vida del ejercito*/
public static String findMaxLifeSoldier(Soldado[][] soldados) { 1 usage new *
    Soldado max = null;
    for (Soldado[] fila : soldados) {
        for (Soldado soldado : fila) {
            /*Verifica que no sean ambos nulo*/
            if (soldado != null && (max == null ||
                soldado.getNivelVida() > max.getNivelVida())) {
                max = soldado;
            }
        }
    }
    return max.toString();
}
```



```
/*Metodo para ordenar el arreglo con bubbleSort en funcion de la vida*/
public static void bubbleSortLife(Soldado[] ejercito) { 1usage new *
    for (int i = 0; i < ejercito.length - 1; i++) {
        for (int j = 0; j < ejercito.length - 1 - i; j++) {
            /*Verifica que no sean ambos nulo*/
            if (ejercito[j] != null && ejercito[j + 1] != null &&
                ejercito[j].getNivelVida() < ejercito[j + 1].getNivelVida()) {
                Soldado temp = ejercito[j];
                ejercito[j] = ejercito[j + 1];
                ejercito[j + 1] = temp;
            }
        }
    }
}

/*Metodo para copiar a un arreglo unidimensional*/
public static Soldado[] toUnidimensional(Soldado[][] soldados){ 1usage new *
    Soldado[] unidimensional = new Soldado[soldados.length*soldados[0].length];
    int index = 0;
    for (Soldado[] fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null) { /*Verifica que no sea nulo*/
                unidimensional[index] = soldado;
                index++;
            }
        }
    }
    return unidimensional;
}

/*Metodo para ordenar el arreglo con insertionSort en funcion de la vida*/
public static void insertionSortLife(Soldado[] ejercito) { 1usage new *
    for (int i = 1; i < ejercito.length; i++) {
        Soldado key = ejercito[i];
        int j = i - 1;
        /*Verifica que no sean ambos nulo*/
        while (j >= 0 && ejercito[j] != null && key != null &&
            ejercito[j].getNivelVida() < key.getNivelVida()) {
            ejercito[j + 1] = ejercito[j];
            j = j - 1;
        }
        ejercito[j + 1] = key;
    }
}
```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

¿Qué resultado esperabas obtener para cada valor de entrada?

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

EJECUCIÓN:

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

```

Soldado de Mayor Vida: [Nombre: Soldado1   Fila: 8 Columna: 2   nivel de Vida: 5]
Promedio de nivel de vida: 2.3333333333333335
Total de vida del ejercito: 14
  
```


Soldados en el orden de creación

```
[Nombre: Soldado0   Fila: 8 Columna: 9   nivel de Vida: 3]
[Nombre: Soldado1   Fila: 8 Columna: 2   nivel de Vida: 5]
[Nombre: Soldado2   Fila: 6 Columna: 10  nivel de Vida: 1]
[Nombre: Soldado3   Fila: 7 Columna: 9   nivel de Vida: 1]
[Nombre: Soldado4   Fila: 2 Columna: 9   nivel de Vida: 1]
[Nombre: Soldado5   Fila: 1 Columna: 5   nivel de Vida: 3]
```



Ranking de poder (Bubble Sort):

```
[Nombre: Soldado1   Fila: 8 Columna: 2   nivel de Vida: 5]
[Nombre: Soldado5   Fila: 1 Columna: 5   nivel de Vida: 3]
[Nombre: Soldado0   Fila: 8 Columna: 9   nivel de Vida: 3]
[Nombre: Soldado4   Fila: 2 Columna: 9   nivel de Vida: 1]
[Nombre: Soldado2   Fila: 6 Columna: 10  nivel de Vida: 1]
[Nombre: Soldado3   Fila: 7 Columna: 9   nivel de Vida: 1]
```

Ranking de poder (Insertion Sort):

```
[Nombre: Soldado1   Fila: 8 Columna: 2   nivel de Vida: 5]
[Nombre: Soldado5   Fila: 1 Columna: 5   nivel de Vida: 3]
[Nombre: Soldado0   Fila: 8 Columna: 9   nivel de Vida: 3]
[Nombre: Soldado4   Fila: 2 Columna: 9   nivel de Vida: 1]
[Nombre: Soldado2   Fila: 6 Columna: 10  nivel de Vida: 1]
[Nombre: Soldado3   Fila: 7 Columna: 9   nivel de Vida: 1]
```

III. CUESTIONARIO:

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



Cambie la rama a main, añadí el informe al área de stage y hice un commit lab05-finalizado y realice el git push -u origin main el origin ya estaba previamente configurado para todos los laboratorios.

LINK:https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS



CONCLUSIONES

El uso de algoritmos de ordenamientos aplicados a arreglos de objetos nos permite realizar búsquedas de los objetos y los atributos correspondientes que necesitamos y nos ayuda a organizar los datos de manera decreciente o creciente de acuerdo a los requisitos que tengamos.

METODOLOGÍA DE TRABAJO

1. Realice la lectura del problema vi lo requisitos y restricciones del mismo para plantear una solución.
2. Identifique las herramientas y la lógica requerida para la solución del mismo.
3. Codifique el problema para luego testarlo con los datos de entrada.
4. Probe la solución y corregí los errores que estuvieron presentes.

REFERENCIAS Y BIBLIOGRAFÍA

	<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;">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: 11</p>

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

E. G. Castro Gutiérrez y M. W. Aedo López, *Fundamentos de programación 2: tópicos de programación orientada a objetos*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021. ISBN: 978-612-5035-20-2. 170 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 20.5 x 29 cm.

Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
--------------------------	--------	-----------	------------	----------

Rubrica:

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	