
	<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: 1</p>

## INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2				
TÍTULO DE LA PRÁCTICA:	Practica de Laboratorio 8: HashMap				
NÚMERO DE PRÁCTICA:	8	AÑO LECTIVO:	2024	NRO. SEMESTRE:	Segundo
FECHA DE PRESENTACIÓN	29/11/2024	HORA DE PRESENTACIÓN	18:20		
INTEGRANTE (s) Santiago Alonso Quintanilla Chávez				NOTA (0-20)	
DOCENTE(s): Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<p><b>I. EJERCICIOS RESUELTOS:</b></p> <ol style="list-style-type: none"> <li>1. Cree un Proyecto llamado Laboratorio8</li> <li>2. Usted deberá crear las dos clases Soldado.java y VideoJuego5.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.</li> <li>3. Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).</li> <li>4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para crear el tablero utilice la estructura de datos más adecuada.</li> <li>5. 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). Se debe mostrar el tablero con todos los soldados creados (distinguir los de un ejército de los del otro ejército). 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 Marco Aedo López 2 ordenamiento de HashMaps). Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo como programa iterativo.</li> </ol>

**a. Código de la clase “Soldado”:**

```
public class Soldado {
    private String nombre;
    private int puntosVida;
    private int fila;
    private int columna;
    private boolean ocupado;
    private int numEjer;
    public Soldado(){
    }
    public void setNombre(String n){
        nombre=n;
    }
    public void setNivelVida(int v) {
        puntosVida=v;
    }
    public void setFila(int f) {
        fila=f;
    }
    public void setColumna (int c) {
        columna=c;
    }
    public void setOcupado (boolean o) {
        ocupado=o;
    }
    public void setNumEjercito (int e){
        numEjer=e;
    }
    public String getNombre(){
        return nombre;
    }
    public int getNivelVida() {
        return puntosVida;
    }
    public int getFila() {
        return fila;
    }
    public int getColumna() {
        return columna;
    }
    public boolean getOcupado() {
        return ocupado;
    }
    public int getNumEjercito(){
        return numEjer;
    }
    public String toString() {
        return "-Nombre: "+nombre+"\t-Posición: (" +fila+", "+columna+")\t-Nivel de Vida: "+puntosVida;
    }
}
```

**b. Código de la clase “Videojuego5”:**

- **Método Principal:** Se utiliza “do-while” para una ejecución iterativa del código, dentro del bucle se usan los métodos creados para ejecutar el programa)

```
import java.util.*;
public class Videojuego5 {
    Run | Debug
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        boolean verificacion=true;
        do {
            HashMap<Integer, HashMap<Integer, Soldado>> tablero=inicializarTablero();
            HashMap<Integer, Soldado> ejercito1=new HashMap<Integer, Soldado>();
            HashMap<Integer, Soldado> ejercito2=new HashMap<Integer, Soldado>();

            inicializarEjercito(tablero, x:1, ejercito1);
            inicializarEjercito(tablero, x:2, ejercito2);
            imprimirTablero(tablero);

            Soldado soldMayor1=soldadoMayorVida(ejercito1);
            Soldado soldMayor2=soldadoMayorVida(ejercito2);
            System.out.println(x:"Los soldados con mayor vida son:");
            System.out.println("\n-Ejercito 1: "+soldMayor1+"\n-Ejercito 2: "+soldMayor2);
            System.out.println(x:"\n_____");
            double vida1=promedioNivelVida(ejercito1);
            double vida2=promedioNivelVida(ejercito2);
            System.out.println("El promedio del nivel de vida del ejercito 1 es: "+vida1);
            System.out.println("El promedio del nivel de vida del ejercito 2 es: "+vida2);
            imprimirOrdenCreacion(ejercito1);
            imprimirOrdenCreacion(ejercito2);
            imprimirRankingVida(ejercito1);
            imprimirRankingVida(ejercito2);
            decidirEjercitoGanador(ejercito1, ejercito2);

            System.out.println(x:"\n_____");
            System.out.println(x:"Desea reiniciar el programa? S/N :");
            String reinicio=scan.next();
            if (reinicio.equals(anObject:"n")||reinicio.equals(anObject:"N")){
                verificacion=false;
            }
        } while(verificacion);
        System.out.println(x:"=====Programa Finalizado=====");
    }
}
```

- **Método inicializarTablero:** crea un tablero vacío bidimensional de 10x10, utilizando un HashMap con claves de tipo Integer y valores de tipo HashMap, que contienen claves de tipo Integer y objetos de tipo Soldado

```
public static void imprimirTablero(HashMap<Integer, HashMap<Integer, Soldado>> arreglo) {
    System.out.println(x: "\n_____");
    System.out.println(x: "Tablero de posiciones: ");
    for (int l=0;l<10;l++){
        System.out.print(s: " ____ ");
    }
    for (int k=0;k<arreglo.size();k++) {
        System.out.println(x: "");
        for (int j=0;j<arreglo.get(k).size();j++) {
            boolean ocupado=arreglo.get(k).get(j).getOcupado();
            if (ocupado) {
                int num=arreglo.get(k).get(j).getNumEjercito();
                System.out.print("|_"+num+"_|");
            } else {
                System.out.print(s: "|__|");
            }
        }
    }
}
```

- **Método inicializarEjercito:** Método que crea un ejército, lo ubica en el mapa (verificando que no se repitan dos soldados en un mismo lugar), y los agrega a un HashMap "ejército"

```
public static void inicializarEjercito(HashMap<Integer, HashMap<Integer, Soldado>> array, int x, HashMap<Integer, Soldado> ejer){
    Random rand=new Random();
    int NumSold=rand.nextInt(bound:10)+1;
    for (int k=0;k<NumSold;k++) {
        int fila=rand.nextInt(bound:10);
        int columna=rand.nextInt(bound:10);
        boolean ocupado=array.get(fila).get(columna).getOcupado();
        while (ocupado) {
            fila=rand.nextInt(bound:10);
            columna=rand.nextInt(bound:10);
            ocupado=array.get(fila).get(columna).getOcupado();
        }
        String nombre="Soldado"+(k)+"x"+(x);
        array.get(fila).get(columna).setNombre(nombre);
        array.get(fila).get(columna).setFila(fila+1);
        array.get(fila).get(columna).setColumna(columna+1);
        array.get(fila).get(columna).setNivelVida(rand.nextInt(bound:5)+1);
        array.get(fila).get(columna).setOcupado(o:true);
        array.get(fila).get(columna).setNumEjercito(x);
        ejer.put(k, array.get(fila).get(columna));
    }
}
```

- **Método imprimirTablero:** Método que imprime el tablero creado, según el formato indicado

```
public static void imprimirTablero(HashMap<Integer, HashMap<Integer, Soldado>> arreglo) {
    System.out.println(x:"\n_____");
    System.out.println(x:"Tablero de posiciones: ");
    for (int l=0;l<10;l++){
        System.out.print(s:" __ ");
    }
    for (int k=0;k<arreglo.size();k++) {
        System.out.println(x:"");
        for (int j=0;j<arreglo.get(k).size();j++) {
            boolean ocupado=arreglo.get(k).get(j).getOcupado();
            if (ocupado) {
                int num=arreglo.get(k).get(j).getNumEjercito();
                System.out.print("|_"+num+"_|");
            } else {
                System.out.print(s:"|__|");
            }
        }
    }
}
```

- **Método soldadoMayorVida:** Método que compara los soldados en un HashMap "Ejército" y devuelve el soldado con mayor vida en el HashMap

```
public static Soldado soldadoMayorVida (HashMap<Integer, Soldado> array) {
    int mayor=0;
    Soldado soldadoMayor=null;
    System.out.println(x:"\n_____");
    for (int i=0;i<array.size();i++) {
        int vida=array.get(i).getNivelVida();
        if (vida>mayor) {
            mayor=vida;
            soldadoMayor=array.get(i);
        }
    }
    return soldadoMayor;
}
```

- **Método promedioNivelVida:** Método que retorna el promedio del nivel de vida de todos los soldados creados de un ejército

```
public static double promedioNivelVida (HashMap<Integer, Soldado> ejer){  
    double acumulado=0.0;  
    for (int i=0;i<ejer.size();i++){  
        double vida=ejer.get(i).getNivelVida();  
        acumulado+=vida;  
    }  
    double promedio=acumulado/(ejer.size());  
    return promedio;  
}
```

- **Método imprimirOrdenCreación:** Método que ordena a los soldados en el orden que fueron creados y los imprime en tal orden--->Se emplea el algoritmo de Ordenamiento Burbuja

```
public static void imprimirOrdenCreacion(HashMap<Integer, Soldado> ejer){  
    System.out.println(x:"\n_____");  
    System.out.println(x:"Orden segun su creacion: ");  
    for (int i=1;i<ejer.size();i++){  
        for (int j=0;j<ejer.size()-1;j++){  
            String ejer_1=ejer.get(j).getNombre();  
            String ejer_2=ejer.get(j+1).getNombre();  
            if ((ejer_1).compareTo(ejer_2)>0){  
                Soldado temp=ejer.get(j);  
                ejer.put(j, ejer.get(j+1));  
                ejer.put(j+1, temp);  
            }  
        }  
    }  
    for (int k=0;k<ejer.size();k++){  
        System.out.println(ejer.get(k));  
    }  
}
```

- **Método imprimirRankingVida:** Método que rankea a los soldados decrecientemente según su nivel de vida y los imprime en tal orden--->Se emplea el algoritmo de Ordenamiento por Selección

```
public static void imprimirRankingVida(HashMap<Integer, Soldado> ejer){  
    System.out.println(x:"\n_____");  
    System.out.println(x:"Ranking del nivel de vida:");  
    int n=0;  
    for (int i=0;i<ejer.size()-1;i++){  
        int mayor=i;  
        n++;  
        for (int j=n;j<ejer.size();j++){  
            if (ejer.get(j).getNivelVida()>ejer.get(mayor).getNivelVida()){  
                mayor=j;  
            }  
        }  
        Soldado temp=ejer.get(mayor);  
        ejer.put(mayor, ejer.get(i));  
        ejer.put(i, temp);  
    }  
    for (int k=0;k<ejer.size();k++){  
        System.out.println(ejer.get(k));  
    }  
}
```

- **Método decidirEjercitoGanador:** Método que compara, primero la vida total del ejército, luego el promedio de vida de cada ejército.
  1. Si en estos dos criterios ocurriese un empate, se compara la cantidad de soldados con vida máxima (5) y se decide el ganador

```
public static void decidirEjercitoGanador(HashMap<Integer, Soldado> ejer1, HashMap<Integer, Soldado> ejer2){
    System.out.println(x: "\n_____");
    int contador_ejer1=0;
    int contador_ejer2=0;
    int acumulado1=0;
    int acumulado2=0;
    for (int i=0;i<ejer1.size();i++){
        acumulado1+=ejer1.get(i).getNivelVida();
    }
    for (int j=0;j<ejer2.size();j++){
        acumulado2+=ejer2.get(j).getNivelVida();
    }
    if (acumulado1>acumulado2){
        contador_ejer1++;
    } if (acumulado2>acumulado1){
        contador_ejer2++;
    }

    if (promedioNivelVida(ejer1)>promedioNivelVida(ejer2)){
        contador_ejer1++;
    } if (promedioNivelVida(ejer2)>promedioNivelVida(ejer1)){
        contador_ejer2++;
    }

    if (contador_ejer1==contador_ejer2){
        int contador1=0;
        int contador2=0;
        for (int s=0;s<ejer1.size();s++){
            if (ejer1.get(s).getNivelVida()==5){
                contador1++;
            }
        }
        for (int r=0;r<ejer2.size();r++){
            if (ejer2.get(r).getNivelVida()==5){
                contador2++;
            }
        }

        if (contador1>contador2){
            contador_ejer1++;
        }
        if (contador2>contador1){
            contador_ejer2++;
        }
    }
    if (contador_ejer1>contador_ejer2){
        System.out.println(x: "El ejercito ganador fue el Ejercito 1!!!");
    } if (contador_ejer2>contador_ejer1) {
        System.out.println(x: "El ejercito ganador fue el Ejercito 2!!!");
    } if (contador_ejer1==contador_ejer2){
        System.out.println(x: "Hubo un empate");
    }
}
```



**c. Ejecución del código:**

-----  
Tablero de posiciones:

		1							
				2				2	
							2		1
								2	
			1		1				
								1	
			1						
1									

-----

-----  
Los soldados con mayor vida son:

-Ejercito 1: -Nombre: Soldado1x1                      -Posición: (6,4)                      -Nivel de Vida: 5  
-Ejercito 2: -Nombre: Soldado3x2                      -Posición: (3,8)                      -Nivel de Vida: 5

-----  
El promedio del nivel de vida del ejercito 1 es: 2.857142857142857  
El promedio del nivel de vida del ejercito 2 es: 3.0

-----  
Orden segun su creacion:

-Nombre: Soldado0x1                      -Posición: (3,10)                      -Nivel de Vida: 3  
-Nombre: Soldado1x1                      -Posición: (6,4)                      -Nivel de Vida: 5  
-Nombre: Soldado2x1                      -Posición: (8,9)                      -Nivel de Vida: 4  
-Nombre: Soldado3x1                      -Posición: (6,6)                      -Nivel de Vida: 1  
-Nombre: Soldado4x1                      -Posición: (1,3)                      -Nivel de Vida: 2  
-Nombre: Soldado5x1                      -Posición: (9,4)                      -Nivel de Vida: 1  
-Nombre: Soldado6x1                      -Posición: (10,1)                      -Nivel de Vida: 4

-----  
Orden segun su creacion:

-Nombre: Soldado0x2                      -Posición: (2,9)                      -Nivel de Vida: 2  
-Nombre: Soldado1x2                      -Posición: (5,9)                      -Nivel de Vida: 2  
-Nombre: Soldado2x2                      -Posición: (2,5)                      -Nivel de Vida: 3  
-Nombre: Soldado3x2                      -Posición: (3,8)                      -Nivel de Vida: 5

-----  
Ranking del nivel de vida:

-Nombre: Soldado1x1	-Posición: (6,4)	-Nivel de Vida: 5
-Nombre: Soldado2x1	-Posición: (8,9)	-Nivel de Vida: 4
-Nombre: Soldado6x1	-Posición: (10,1)	-Nivel de Vida: 4
-Nombre: Soldado0x1	-Posición: (3,10)	-Nivel de Vida: 3
-Nombre: Soldado4x1	-Posición: (1,3)	-Nivel de Vida: 2
-Nombre: Soldado5x1	-Posición: (9,4)	-Nivel de Vida: 1
-Nombre: Soldado3x1	-Posición: (6,6)	-Nivel de Vida: 1

-----  
Ranking del nivel de vida:

-Nombre: Soldado3x2	-Posición: (3,8)	-Nivel de Vida: 5
-Nombre: Soldado2x2	-Posición: (2,5)	-Nivel de Vida: 3
-Nombre: Soldado1x2	-Posición: (5,9)	-Nivel de Vida: 2
-Nombre: Soldado0x2	-Posición: (2,9)	-Nivel de Vida: 2

-----  
Hubo un empate

-----  
Desea reiniciar el programa? S/N :

n

=====Programa Finalizado=====

PS C:\Users\PC\Documents\FP\_2\Laboratorio08> |

- **Evidencia de los Commits:**
  - **Commits del Laboratorio del día 26 de noviembre:**

Commit changes

Commit message

Create Soldado

Extended description

Add an optional extended description..

☒ Commit directly to the main branch
 ☐ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

Cancel

Commit changes

10:05:59 a. m.

martes, 26 de noviembre de 2024

noviembre de 2024

do.	lu.	ma.	mi.	ju.	vi.	sá.
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Hoy

Configura los calendarios para ver

Commit changes

Commit message

Create Videojuego5

Extended description

Add an optional extended description..

☒ Commit directly to the main branch
 ☐ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

Cancel

Commit changes

10:07:00 a. m.

martes, 26 de noviembre de 2024

noviembre de 2024

do.	lu.	ma.	mi.	ju.	vi.	sá.
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Hoy

Configura los calendarios para ver

○ **Commits del Laboratorio del día 29 de noviembre:**

Commit changes

Commit message

Update Soldado

Extended description

Add an optional extended description..

☒ Commit directly to the main branch
 ☐ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

Cancel

Commit changes

18:06:19

viernes, 29 de noviembre de 2024

Noviembre de 2024

DO	LU	MA	MI	JU	VI	SA
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Hoy

Configura los calendarios para ver

Commit changes

Commit message

Update Videojuego5

Extended description

Add an optional extended description..

☒ Commit directly to the main branch
 ☐ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

Cancel

Commit changes

18:05:22

viernes, 29 de noviembre de 2024

Noviembre de 2024

DO	LU	MA	MI	JU	VI	SA
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Hoy

Configura los calendarios para ver

- **Link al repositorio de GitHub - Laboratorio05:**  
[https://github.com/SantiagoQuintanilla/LaboratorioFP2/tree/main/LABORATORIO\\_08](https://github.com/SantiagoQuintanilla/LaboratorioFP2/tree/main/LABORATORIO_08)

● **Autoevaluación:**

Contenido y demostración		Puntos	Checklist	Estudiante	Pr
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	1	
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		17	

Tabla 2: Rúbrica para contenido del Informe y demostración