# Informe de Laboratorio 08

## Tema: Laboratorio 08

| Nota |
|---|
| |

| Estudiante | Escuela | Asignatura |
|---|---|---|
| Victor Mamani Anahua<br>vmamanian@unsa.edu.pe | Escuela Profesional de Ingeniería de Sistemas | Fundamentos de la Programación II<br>Semestre: II<br>Código: 20230489 |

| Laboratorio | Tema | Duración |
|---|---|---|
| 08 | Laboratorio 08 | 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 Laboratorio8
- Usted deberá crear las dos clases Soldado.java y VideoJuego5.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 (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).
- 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 (indicar conclusiones respecto a este ordenamiento de HashMaps).
- 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 08.

# 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/VictorMA18/fp2-23b.git`
- URL para el laboratorio 08 en el Repositorio GitHub.
- `https://github.com/VictorMA18/fp2-23b/tree/main/Fase02/Lab08`

# 4. Actividades del Laboratorio 08

## 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 VideoJuego5.
- El codigo y el commit seria el siguiente:

Listing 1: Commit

```
$ git commit -m "Publicando la clase Soldado para su posterior uso en el siguiente
    ejercicio que es con Hashmaps"
```

Listing 2: Las lineas de codigos del metodo creado:

```
// Laboratorio Nro 8 - 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 VideoJuego5

- En el segundo commit creamos el metodo mapHashFillRegister() el cual usamos el uso de HashMaps y tambien de ArrayList para poder añadirlos al hashmaps ya que en este hariamos la comprobacion de que cada Soldado gracias a una iteracion el cual este comprueba que estos si son nulos se añadiria al soldado en la cuadrilla correspondiente y si no no lo añadiria y retrocederia una iteracion ya que se contaria por lo que nos lleva a que en este no se repita en misma cuadrilla despues de esto imprimiriamos los datos del soldado en el orden que fueron creados

Listing 3: Commit

```
$ git commit -m "Creando el metodo mapHashFillRegister() el cual usamos el uso de
    HashMaps y tambien de ArrayList para poder anadirlos al hashmaps ya que en este
    hariamos la comprobacion de que cada Soldado en este no se repita en misma
    cuadrilla despues de esto imprimiriamos los datos del soldado en el orden que
    fueron creados"
```

Listing 4: Las lineas de codigos del metodo creado:

```
// Laboratorio Nro 8 - Ejercicio VideoJuego5
// Autor: Mamani Anahua Victor Narciso
// Colaboro:
// Tiempo:
import java.util.*;
class VideoJuego5{
   public static HashMap<String, Soldado> mapHashFillRegister(int num){
      Random rdm =new Random();
      HashMap<String, Soldado> army1 = new HashMap<String, Soldado>();
      ArrayList<ArrayList<Soldado>> army = new ArrayList<ArrayList<Soldado>>(); //NOS
          AYUDARIAMOS DE UN ARRAYLIST PARA PODER AYUDARNOS CON EL USO DE HASHMAPS PARA
          PODER REGISTAR A SOLDADOS EN LA QUE NINGUNO DE ESTOS SE REPITA
      int numsoldiers = rdm.nextInt(10) + 1; //NUMERO DE SOLDADOS QUE SE VAN A CREAR DE
          1 AL 10
      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 " + numsoldiers + " soldados : "
          );
      System.out.println("**********************************");
      for(int i = 0; i < numsoldiers; i++){ //ITERACION PARA PODER DARLES LOS DATOS A
          CADA SOLDADO CREADO
         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 VIDEOJUEGO4.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));
            army1.put("Soldado" + i, new Soldado(name, health, row, column));
                //INTEGRAMOS AL HASHMAP AL SOLDADO CON SU RESPECTIVO NOMBRE Y VALOR
            System.out.println(army1.get("Soldado"+ i).toString()); //PUBLICAMOS AL
                SOLDADO CREADO POR ORDEN DE CREACION
         }else{
            i -= 1; //NOS AYUDARIA CON LOS SOLDADOS QUE SE REPITEN EN EL MISMO CASILLERO
```

```
                    CON TAL QUE NO DEBERIA CONTAR
                }
            }
            return army1;
        }
        public static void main (String args []){
            HashMap<String, Soldado> army1 = mapHashFillRegister(1);
            HashMap<String, Soldado> army2 = mapHashFillRegister(2);
        }
    }
```

Listing 5: Ejecucion:

```
El Ejercito 1 tiene 4 soldados :
*********************************
Registrando al 1 soldado del Ejercito 1
------------------
Nombre: Soldado0X1
Vida: 5
Fila: 4
Columna: D
Registrando al 2 soldado del Ejercito 1
------------------
Nombre: Soldado1X1
Vida: 3
Fila: 8
Columna: C
Registrando al 3 soldado del Ejercito 1
------------------
Nombre: Soldado2X1
Vida: 5
Fila: 6
Columna: C
Registrando al 4 soldado del Ejercito 1
------------------
Nombre: Soldado3X1
Vida: 5
Fila: 8
Columna: I
El Ejercito 2 tiene 1 soldados :
*********************************
Registrando al 1 soldado del Ejercito 2
------------------
Nombre: Soldado0X2
Vida: 5
Fila: 2
Columna: B
```

## 4.3. Estructura de laboratorio 08

- El contenido que se entrega en este laboratorio08 es el siguiente:

```
/Lab08
   "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/1bKx0eCXY6JlkP_RaiD2uaH1JQzoSn6uN/view