# Manual Técnico

# Arena USAC

**Introducción a la programación y computación 1**

**Katherin Yasmin Miranda Hernández**

**Carné: 202300537**

# Introducción

El presente manual técnico describe la implementación del sistema Arena USAC, una aplicación desarrollada en el lenguaje Java que permite simular batallas entre dos personajes.

El programa utiliza una interfaz gráfica construida con Swing y hace uso de hilos para ejecutar las batallas simultáneamente. Además, se incluye la persistencia de datos mediante archivos de texto para guardar y cargar los personajes creados.

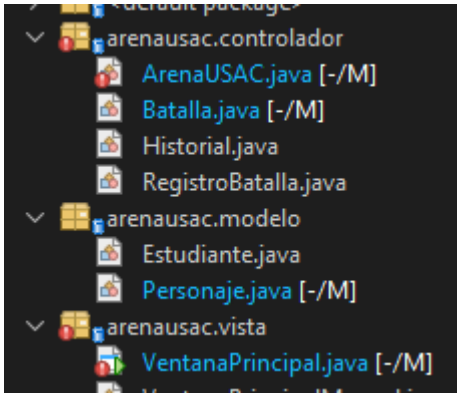
# Requerimientos:

- Lenguaje: Java 8 o superior
- Entorno de desarrollo: NetBeans IDE 12 o superior
- Librerías utilizadas: Swing
- Sistema operativo: Windows, Linux
- Memoria: 4gb de RAM como mínimo

# Arquitectura del sistema

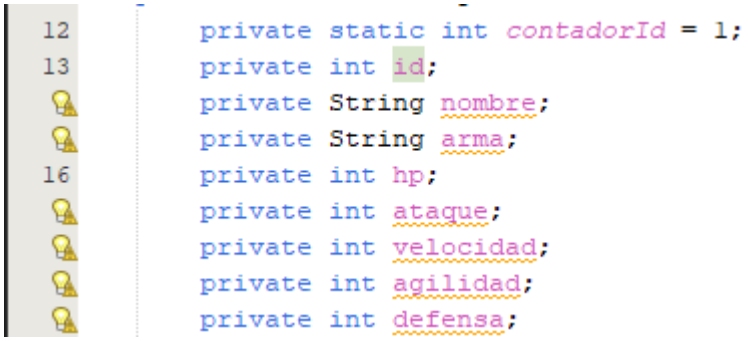
El sistema está basado en el patrón MVC (Modelo-Vista-Controlador):

- Modelo: contiene las clases Personaje, Estudiante.
- Vista: contiene la VentanaPrincipal, diseñada con la GUI de NetBeans.
- Controlador: maneja la lógica del programa con las clases ArenaUSAC, Batalla, Historial y RegistroBatalla.



Clases:

- **Personaje:**
  Esta clase representa a un combatiente en la arena
  Sus atributos principales son: id, nombre, arma, hp, ataque, velocidad, agilidad y defensa.




  Sus métodos principales son: getHp(), setH(), getAtaque(), y setId()

```java
    public String getNombre() {
        return nombre; }

    public String getArma() {
        return arma;
    }
    public int getHp() {
        return hp;
    }
    public int getAtaque() {
        return ataque;
    }
    public int getVelocidad() {
        return velocidad;
    }
    public int getAgilidad() {
        return agilidad;
    }
    public int getDefensa() {
        return defensa;
    }

    public void setHp(int hp){
        this.hp = hp;
    }

    public void setId(int id){
        this.id =id;
    }

    public int getId(){
        return id;
    }
```

- **ArenaUSAC:**
  Esta clase es la encargada de controlar el registro de personajes, y la carga desde archivos
  Sus atributos son: Personaje[] personajes, int cantidad

```java
public class ArenaUSAC {
    private Personaje[] personajes;
    private int cantidad;
```

  Sus métodos son: agregarPersonaje(), guardarPersonajes(), cargarPersonajes(), modificarPersonaje(), buscarPorId()

```java
public void modificarPersonaje(int id, String arma, int hp, int ataque, int velocidad, int agilidad, int defensa){
    for (int i = 0; i < cantidad; i++){
        if (personajes[i].getId() == id){
            personajes[i] = new Personaje(personajes[i].getNombre(),
                    arma, hp, ataque, velocidad, agilidad, defensa);{

            return;
            }
        }
        System.out.println("No se encontro el personaje con ID " + id);
    }
}
```

```java
public void agregarPersonaje(Personaje p){
    if (cantidad >= personajes.length){
        System.out.println("Error, no se pueden agregar mas personajes");
        return;
    }

    for (int i = 0; i< cantidad; i++){
        if (personajes[i].getNombre().equalsIgnoreCase(p.getNombre())){
            System.out.println("Error, ya existe un personaje con ese nombre");
            return;
        }
    }

    personajes[cantidad] = p;
    cantidad++;
    System.out.println("Personaje agregado: " + p.getNombre());
}
```

```java
public Personaje buscarPorId(int id){
    for (int i = 0; i < cantidad; i++){
        if (personajes[i].getId() == id){
            return personajes[i];
        }
    }
    return null;
}
```

- **Batalla:**
  Esta clase se encarga de controlar la ejecución de cada batalla entre dos personajes, también utiliza hilos para mostrar el porcentaje de vida restante de cada personaje

```java
public class Batalla {
    private Personaje p1;
    private Personaje p2;
    private Historial historial;
    private JTextArea areaTexto;
    private boolean enCurso;

    private JProgressBar barraP1;
    private JProgressBar barraP2;
```

```
46      public Personaje iniciar() {
47          Thread t1 = new Thread(() -> atacar(p1, p2, barraP2));
48          Thread t2 = new Thread(() -> atacar(p2, p1, barraP1));
49
50          t1.start();
51          t2.start();
52
53          try {
54              t1.join(); // espera a que termine t1
55              t2.join(); // espera a que termine t2
56          } catch (InterruptedException e) {
                e.printStackTrace();
58          }
59
60          // Determinar ganador
61          if (p1.getHp() > 0) {
62              return p1;
63          } else {
64              return p2;
65          }
66      }
```

- **Estudiante:**
  Es la clase encargada de gestionar los datos del estudiante

```
11  public class Estudiante {
        private String nombre;
        private String carne;
14
15
16      public Estudiante(String nombre, String carne) {
17          this.nombre = nombre;
18          this.carne = carne;
19
20      }
21
22      @Override
        public String toString() {
24          return "DATOS DEL ESTUDIANTE" +
25                  "Nombre: " + nombre + "\n" +
26                  "Carne: " + carne + "\n";
27      }
28  }
```

- **VentanaPrincipal:**
  Contiene todos los elementos visuales como los botones, labels y barras
  Entre sus eventos mas importantes se encuentran los siguientes

btnAgregarPersonajeActionPerformed() que es la encargada de agregar personajes

```
316     private void btnAgregarPersonajeActionPerformed(java.awt.event.ActionEvent evt) {
317         // TODO add your handling code here:
318         try {
319             String nombre = JOptionPane.showInputDialog(this, "Nombre:");
320             String arma = JOptionPane.showInputDialog(this, "Arma:");
321             int hp = Integer.parseInt(JOptionPane.showInputDialog(this, "HP (100-500):"));
322             int ataque = Integer.parseInt(JOptionPane.showInputDialog(this, "Ataque (10-100):"));
323             int velocidad = Integer.parseInt(JOptionPane.showInputDialog(this, "Velocidad (1-10):"));
324             int agilidad = Integer.parseInt(JOptionPane.showInputDialog(this, "Agilidad (1-10):"));
325             int defensa = Integer.parseInt(JOptionPane.showInputDialog(this, "Defensa (1-50):"));
326 
327             Personaje p = new Personaje(nombre, arma, hp, ataque, velocidad, agilidad, defensa);
328             arena.agregarPersonaje(p);
329             JOptionPane.showMessageDialog(this, "Personaje agregado");
331         } catch (Exception ex) {
332             JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
333         }
    }
```

btnIniciarBatallaActionPerformed() que es la que inicia la batalla

```
418     private void btnIniciarBatallaActionPerformed(java.awt.event.ActionEvent evt) {
419         // TODO add your handling code here:
420         int id1 = Integer.parseInt(JOptionPane.showInputDialog(this, "ID del primer personaje:"));
421         int id2 = Integer.parseInt(JOptionPane.showInputDialog(this, "ID del segundo personaje:"));
422         Personaje p1 = arena.buscarPorId(id1);
423         Personaje p2 = arena.buscarPorId(id2);
424 
425         if (p1 != null && p2 != null) {
426             actualizarNombresBarras(p1,p2);
427 
428             new Thread(() -> {
429                 Batalla batalla = new Batalla(p1, p2, historial, new JTextArea(), barraP1, barraP2);
430                 Personaje ganador = batalla.iniciar();
431 
432                 SwingUtilities.invokeLater(() -> {
433                 JOptionPane.showMessageDialog(
434                     this,
435                     "¡El ganador es: " + ganador.getNombre() + "!",
436                     "Resultado de la batalla",
437                     JOptionPane.INFORMATION_MESSAGE
438                 );
439             });
440 
441             }).start();
442             JOptionPane.showMessageDialog(this, "Batalla iniciada");
443         } else {
444             JOptionPane.showMessageDialog(this, "Alguno de los personajes no existe");
445         }
446     }
```

btnCargarPersonajesActionPerformed() que es la que los carga

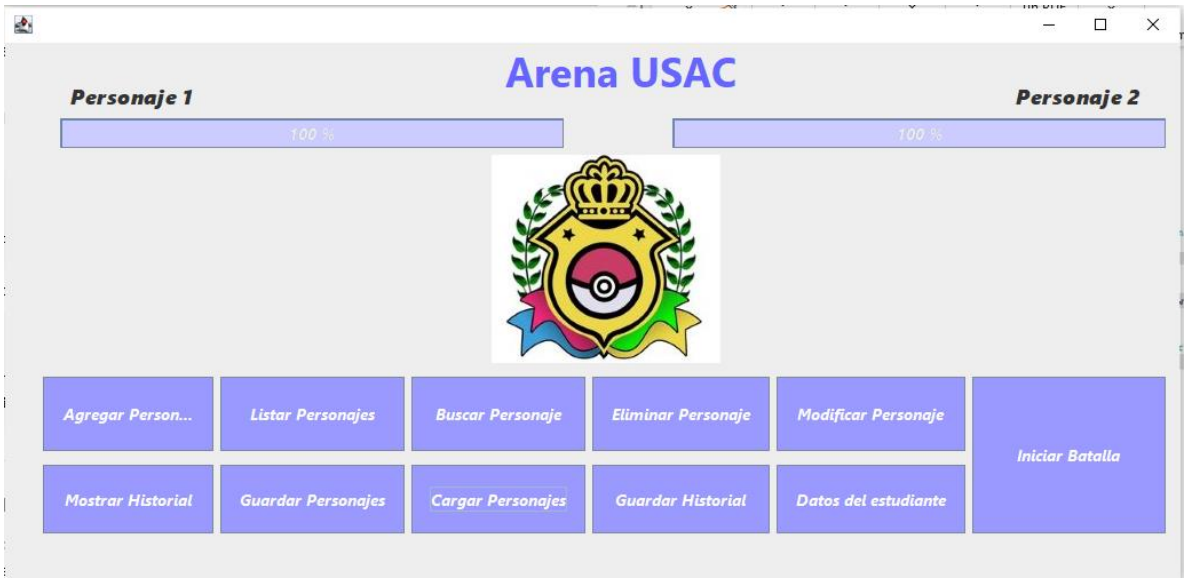
```
399     private void btnCargarPersonajesActionPerformed(java.awt.event.ActionEvent evt) {
400         // TODO add your handling code here:
401         arena.cargarPersonajes("personajes.txt");
402         JOptionPane.showMessageDialog(this, "Personajes cargados");
403     }
404 
```

## Descripción de la interfaz gráfica

La interfaz gráfica fue diseñada con el editor visual de NetBeans.

Incluye:

- Un titulo con el logo Arena Usac
- Dos barras de progreso que representan la vida de cada personaje, así mismo, cuenta con etiquetas para identificar cada barra con el nombre del personaje
- Botones para las diferentes acciones que emplea el programa
- Ventana emergente para mostrar el ganador de la batalla



# Diagrama de flujo