

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
LENGUAJES FORMALES DE PROGRAMACION
CATEDRÁTICO: ING. DAMARIS CAMPOS
TUTOR ACADÉMICO: LUISA MARIA ORTIZ



MANUAL TÉCNICO **PRACTICA 1**

Álvaro Gabriel Ceballos Gil
CARNÉ: 202300673
SECCIÓN: A-

GUATEMALA, 02 DE MARZO DEL 2,025

ÍNDICE

ÍNDICE	1
INTRODUCCIÓN	1
OBJETIVOS	3
1. GENERAL	3
2. ESPECÍFICOS	3
ALCANCES DEL SISTEMA	4
ESPECIFICACIÓN TÉCNICA	5
• REQUISITOS DE HARDWARE	6
• REQUISITOS DE SOFTWARE	6
IMPORTACION DE LIBRERIAS	7
MENU PRINCIPAL	7
Clase PersonajeEnJuego	8
Procedimiento LeerArchivo	8
Procedimiento LeerPersonajes	9
Procedimiento reporteAtaque	10
Procedimiento reporteDefensa	10
Procedimiento startbattle	12

INTRODUCCIÓN

Este manual se encontrará de forma más desarrollada la explicación del código del proyecto, con el fin de que un programador pueda entender el código del programa, y así poder implementarlo si así lo desea.

Este manual le permite al programador ver la lógica del programa de una forma mucho más comprensible, lo que permitirá un mejor desarrollo de la aplicación en el caso de que el programador desee implementar alguna función de este programa en el suyo.

OBJETIVOS

1. GENERAL

- 1.1. Ayudar al programador a poder entender el programa de una mejor manera

2. ESPECÍFICOS

- 2.1. Explicar la lógica de las funcionalidades del programa de una forma más sencilla
- 2.2. Brindar información necesaria para la comprensión del proyecto de una forma más técnica

ALCANCES DEL SISTEMA

Este manual tiene el objetivo de explicar de una forma mucho más explícita las funcionalidades del código de la practica 1, con el fin de que el programador sea capaz de entender correctamente las líneas de código utilizadas para el desarrollo del programa.

También se tiene como objetivo que el programador logre entender mejor la lógica del programa, y que de esta forma el programador sea capaz de implementar varias funcionalidades de este programa en su código.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**

Para poder correr este programa, es necesario que el usuario tenga instalado java con NetBeans para el correcto desarrollo de la practica

Se debe de contar con un sistema operativo Windows de 64 bits, con una memoria RAM recomendada de 8GB, ya que con esto se garantiza de que el programa pueda correr sin ningún tipo de dificultad.

- **REQUISITOS DE SOFTWARE**

- Debe contar con un sistema operativo medianamente moderno, como Windows 10, esto con el fin de que a la hora de desarrollar el proyecto no tenga algún tipo de error.

EXPLICACION DEL CODIGO

IMPORTACION DE LIBRERIAS

```

*
* @author aceba
*/
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;
import java.util.Scanner;
import java.util.Stack;
import javax.swing.*;
import main.Personaje;
```

Para el correcto funcionamiento del programa y para una mayor optimización de código

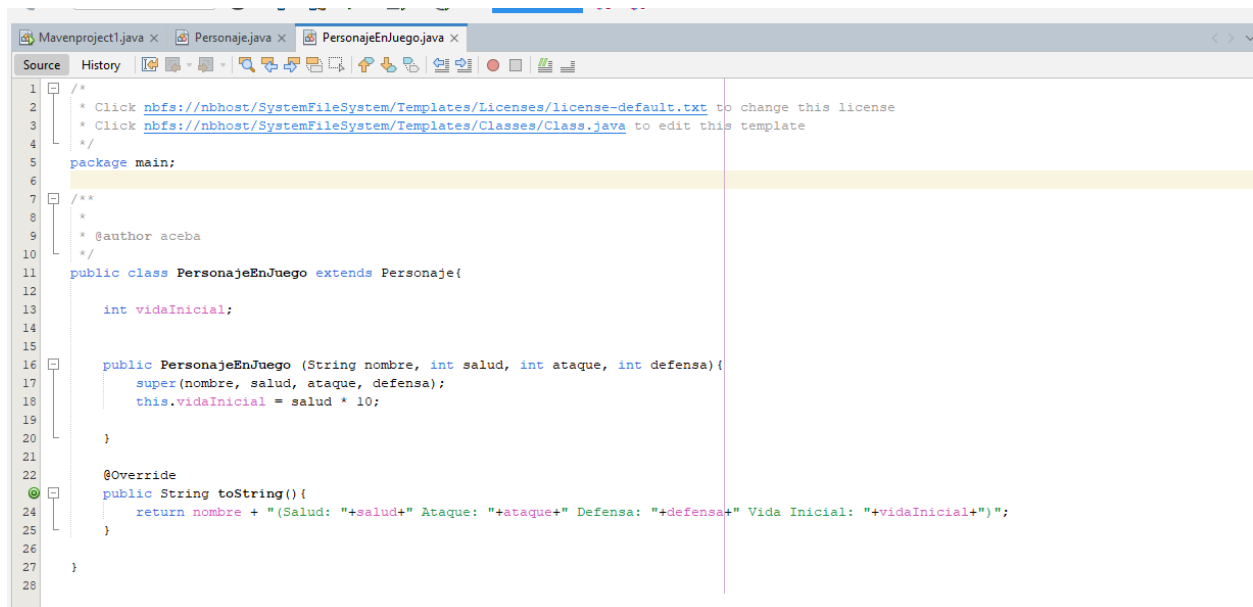
MENU PRINCIPAL

Para hacer el menú principal se hizo uso de un do while, con un switch and case, hola donde cada opción se representa en dicho ciclo de manera

```
List<PersonajeEnJuego> personajes = new ArrayList<>();
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int option;
    Mavenproject1 lfpaTarea2_202300673 = new Mavenproject1();
    do{
        System.out.println("Menu principal");
        System.out.println("1. Cargar archivo");
        System.out.println("2. Jugar");
        System.out.println("3. Generar reporte de mayor ataque");
        System.out.println("4. Generar reporte con mayor defensa");
        System.out.println("5. Mostrar informacion del desarrollador");
        System.out.println("6. Salir");
        option = scanner.nextInt();
        switch (option){
            case 1:
                lfpaTarea2_202300673.lecturaArchivo();
                break;
            case 2:
                lfpaTarea2_202300673.startBattle();
                break;
            case 3:
                lfpaTarea2_202300673.reporteAtaque();
                break;
            case 4:
                lfpaTarea2_202300673.reporteDefensa();
                break;
            case 5:
                System.out.println("Nombre: Alvaro Gabriel Ceballos Gil");
                System.out.println("Carnet: 202300673");
                System.out.println("Lenguajes Formales de Programacion, seccion A-");
                break;
            default:
                System.out.println("Has cerrado el programa");
                break;
        }
    }
}

}while(option!=6);
```

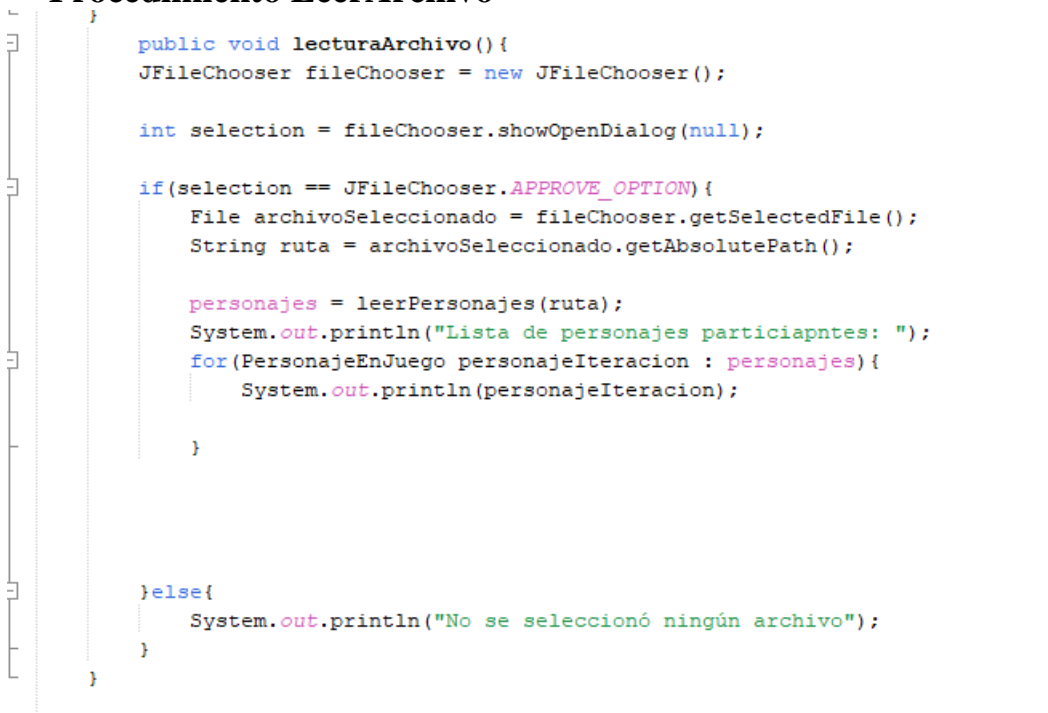
Clase PersonajeEnJuego



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package main;
6
7  /**
8   *
9   * @author aceba
10  */
11  public class PersonajeEnJuego extends Personaje{
12
13      int vidaInicial;
14
15
16      public PersonajeEnJuego (String nombre, int salud, int ataque, int defensa){
17          super(nombre, salud, ataque, defensa);
18          this.vidaInicial = salud * 10;
19      }
20
21
22      @Override
23      public String toString(){
24          return nombre + " (Salud: "+salud+" Ataque: "+ataque+" Defensa: "+defensa+" Vida Inicial: "+vidaInicial+");";
25      }
26
27  }
28
```

O se puede observar en el código esta clase es una herencia de otra llamada personaje ya que en otra clase se definieron los atributos como la salud la defensa en ataque y el hombre y en esto se Añade la vida inicial. Hoy se coloca su constructor con un tu string que dirá los datos iniciales para verificar que se haya cargado correctamente a la hora de importar

Procedimiento LeerArchivo



```
1  }
2
3  public void lecturaArchivo(){
4      JFileChooser fileChooser = new JFileChooser();
5
6      int selection = fileChooser.showOpenDialog(null);
7
8      if(selection == JFileChooser.APPROVE_OPTION){
9          File archivoSeleccionado = fileChooser.getSelectedFile();
10         String ruta = archivoSeleccionado.getAbsolutePath();
11
12         personajes = leerPersonajes(ruta);
13         System.out.println("Lista de personajes particiapntes: ");
14         for(PersonajeEnJuego personajeIteracion : personajes){
15             System.out.println(personajeIteracion);
16         }
17     }
18     else{
19         System.out.println("No se seleccionó ningún archivo");
20     }
21 }
22
```

En dicha clase se hace todo el procedimiento necesario para que se pueda leer cualquier tipo de archivo de extensión el FP donde se hace un foreach sobre una lista que tiene la clase de personaje en juego, no olvidar de siempre quedar nuestras

listas para poder ir almacenando nuestros datos.

Procedimiento LeerPersonajes

```
public List<PersonajeEnJuego> leerPersonajes(String ruta){
    List<PersonajeEnJuego> personajeLista2 = new ArrayList<>();

    try (BufferedReader br = new BufferedReader(new FileReader(ruta))){

        String linea;
        br.readLine();//Leyendo los encabezados

        while((linea = br.readLine())!=null){
            String[] datos = linea.split("\\|");

            //Encontrando los datos
            String nombre = datos[0].trim();
            int salud = Integer.parseInt(datos[1].trim());
            int ataque = Integer.parseInt(datos[2].trim());
            int defensa = Integer.parseInt(datos[3].trim());

            PersonajeEnJuego personajeJuego = new PersonajeEnJuego(nombre, salud, ataque, defensa);
            personajeLista2.add(personajeJuego);
        }

    } catch (IOException e){
        System.out.println("Error al leer el archivo: "+e.getMessage());
    }

    return personajeLista2;
}
```

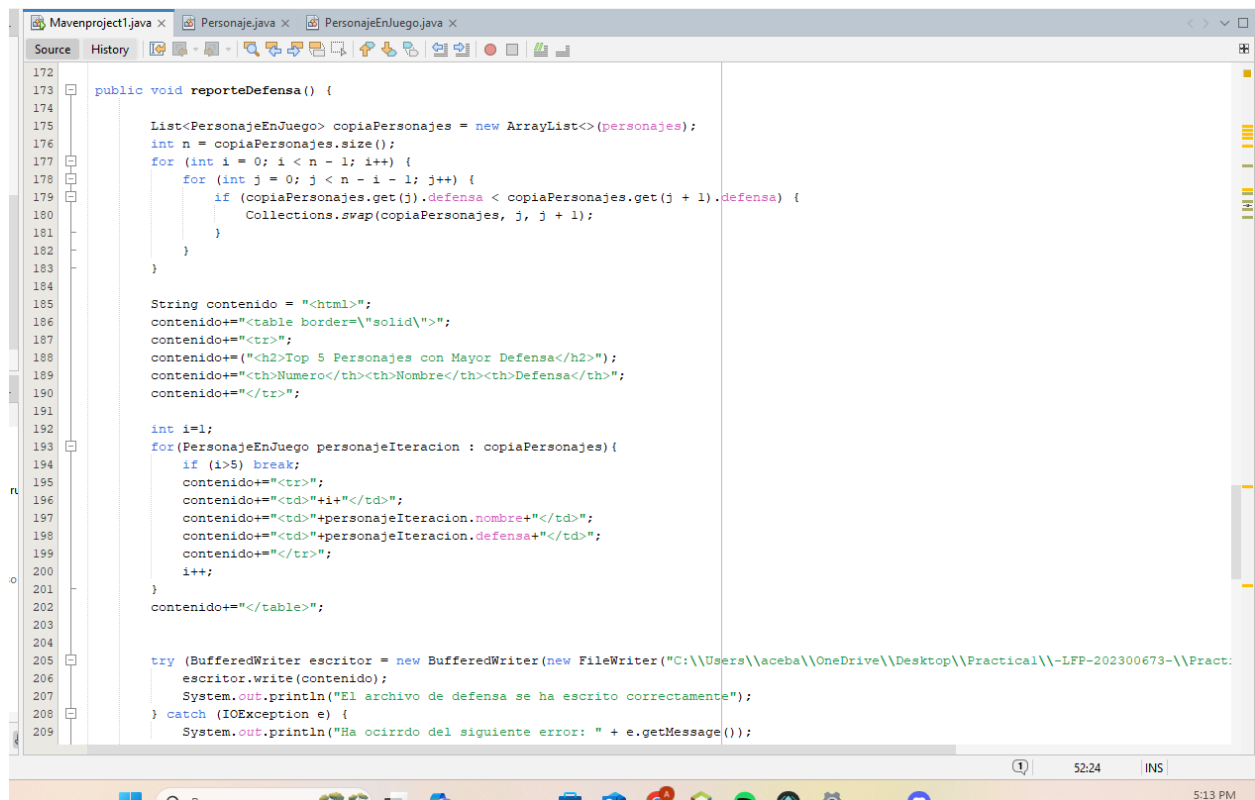
En dicho procedimiento se crea una segunda lista, hoy se hace uso del buffer reader para poder leer los elementos extraídos en el procedimiento anterior, donde se hace un split para poder ir separando los elementos que venían separados e irlos almacenando dentro de datos, para luego estos datos irlos agregando en nuestra lista de personaje en juego. Al final se retorna a la lista ya con todos los datos añadidos.

Procedimiento reporteAtaque

```
public void reporteAtaque() {  
    List<PersonajeEnJuego> copiaPersonajes = new ArrayList<>(personajes);  
    int n = copiaPersonajes.size();  
    for (int i = 0; i < n - 1; i++) {  
        for (int j = 0; j < n - i - 1; j++) {  
            if (copiaPersonajes.get(j).ataque < copiaPersonajes.get(j + 1).ataque) {  
                Collections.swap(copiaPersonajes, j, j + 1);  
            }  
        }  
    }  
  
    String contenido = "<html>";  
    contenido+="<table border=\"solid\">";  
    contenido+="<tr>";  
    contenido+=("<h2>Top 5 Personajes con Mayor Ataque</h2>");  
    contenido+="<th>Numero</th><th>Nombre</th><th>Ataque</th>";  
    contenido+="</tr>";  
  
    int i=1;  
    for(PersonajeEnJuego personajeIteracion : copiaPersonajes){  
        if (i>5) break;  
        contenido+="<tr>";  
        contenido+="<td>"+i+"</td>";  
        contenido+="<td>"+personajeIteracion.nombre+"</td>";  
        contenido+="<td>"+personajeIteracion.ataque+"</td>";  
        contenido+="</tr>";  
        i++;  
    }  
  
    contenido+="</table>";  
  
    try (BufferedWriter escritor = new BufferedWriter(new FileWriter("C:\\Users\\aceba\\OneDrive\\Desktop\\Practical\\-LFP-202300673-\\Pract:")) {  
        escritor.write(contenido);  
    }  
}
```

En este procedimiento se construye el archivo html, inicialmente se hace el uso de un ordenamiento por burbuja donde se van intercalando los elementos y se van alineando conforme a lo solicitado, luego se genera la estructura de un archivo html donde se coloca que queremos en las columnas y filas para de último almacenarlo dentro de una carpeta y poder visualizar dicho archivo creado de formato html.

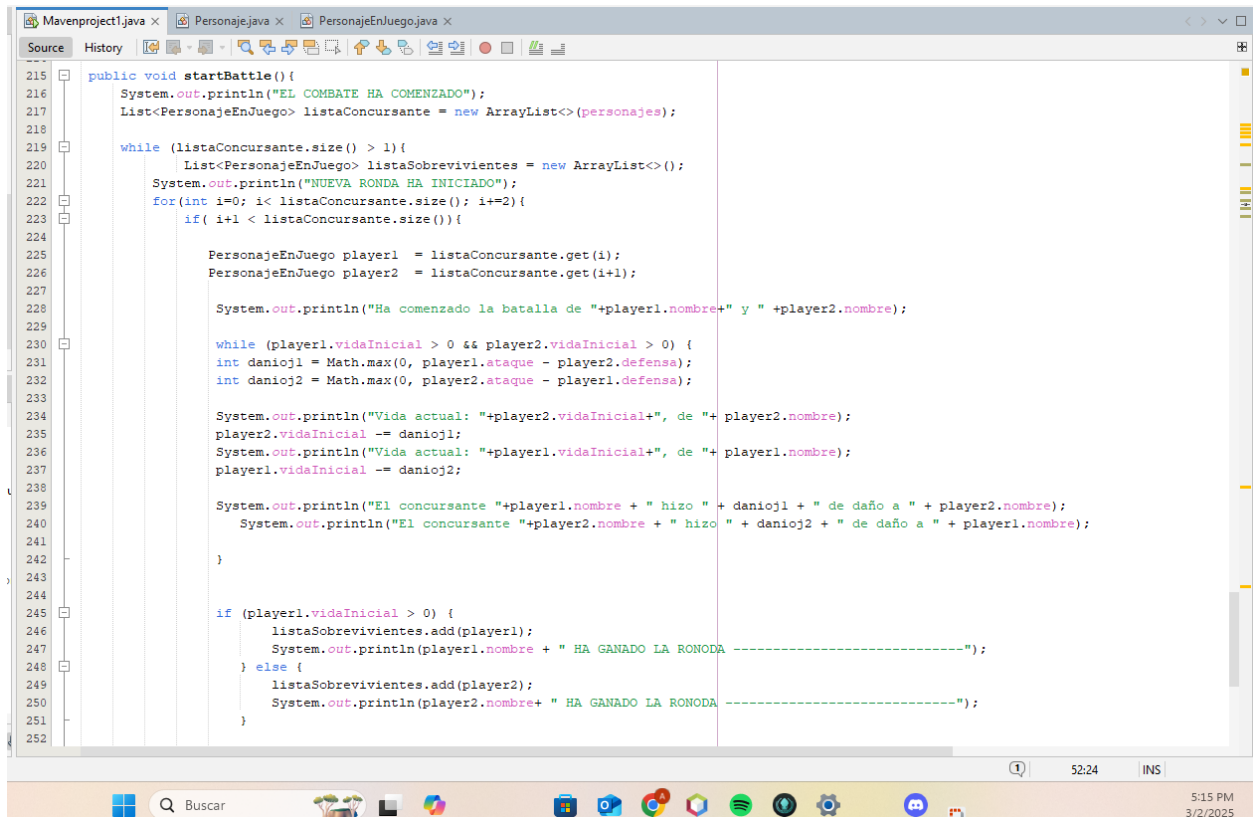
Procedimiento reporteDefensa



```
172
173 public void reporteDefensa() {
174
175     List<PersonajeEnJuego> copiaPersonajes = new ArrayList<>(personajes);
176     int n = copiaPersonajes.size();
177     for (int i = 0; i < n - 1; i++) {
178         for (int j = 0; j < n - i - 1; j++) {
179             if (copiaPersonajes.get(j).defensa < copiaPersonajes.get(j + 1).defensa) {
180                 Collections.swap(copiaPersonajes, j, j + 1);
181             }
182         }
183     }
184
185     String contenido = "<html>";
186     contenido+="<table border=\"solid\">";
187     contenido+="<tr>";
188     contenido+="(<h2>Top 5 Personajes con Mayor Defensa</h2>);
189     contenido+="<th>Numero</th><th>Nombre</th><th>Defensa</th>";
190     contenido+="</tr>";
191
192     int i=1;
193     for(PersonajeEnJuego personajeIteracion : copiaPersonajes){
194         if (i>5) break;
195         contenido+="<tr>";
196         contenido+="<td>"+i+"</td>";
197         contenido+="<td>"+personajeIteracion.nombre+"</td>";
198         contenido+="<td>"+personajeIteracion.defensa+"</td>";
199         contenido+="</tr>";
200         i++;
201     }
202     contenido+="</table>";
203
204
205     try (BufferedWriter escritor = new BufferedWriter(new FileWriter("C:\\Users\\aceba\\OneDrive\\Desktop\\Practica1\\-LFP-202300673-\\Pract:
206         escritor.write(contenido);
207         System.out.println("El archivo de defensa se ha escrito correctamente");
208     } catch (IOException e) {
209         System.out.println("Ha ocurrido del siguiente error: " + e.getMessage());
```

En este procedimiento se construye el archivo html, inicialmente se hace el uso de un ordenamiento por burbuja donde se van intercalando los elementos y se van alineando conforme a lo solicitado, luego se genera la estructura de un archivo html donde se coloca que queremos en las columnas y filas para de último almacenarlo dentro de una carpeta y poder visualizar dicho archivo creado de top de defensa de formato html de forma correcta.

Procedimiento startbattle



```
215 public void startBattle(){
216     System.out.println("EL COMBATE HA COMENZADO");
217     List<PersonajeEnJuego> listaConcursante = new ArrayList<>(personajes);
218
219     while (listaConcursante.size() > 1){
220         List<PersonajeEnJuego> listaSobrevivientes = new ArrayList<>();
221         System.out.println("NUEVA RONDA HA INICIADO");
222         for(int i=0; i< listaConcursante.size(); i+=2){
223             if( i+1 < listaConcursante.size()){
224
225                 PersonajeEnJuego player1 = listaConcursante.get(i);
226                 PersonajeEnJuego player2 = listaConcursante.get(i+1);
227
228                 System.out.println("Ha comenzado la batalla de "+player1.nombre+" y "+player2.nombre);
229
230                 while (player1.vidaInicial > 0 && player2.vidaInicial > 0) {
231                     int danioj1 = Math.max(0, player1.ataque - player2.defensa);
232                     int danioj2 = Math.max(0, player2.ataque - player1.defensa);
233
234                     System.out.println("Vida actual: "+player2.vidaInicial+", de "+ player2.nombre);
235                     player2.vidaInicial -= danioj1;
236                     System.out.println("Vida actual: "+player1.vidaInicial+", de "+ player1.nombre);
237                     player1.vidaInicial -= danioj2;
238
239                     System.out.println("El concursante "+player1.nombre + " hizo " + danioj1 + " de daño a " + player2.nombre);
240                     System.out.println("El concursante "+player2.nombre + " hizo " + danioj2 + " de daño a " + player1.nombre);
241                 }
242
243
244
245                 if (player1.vidaInicial > 0) {
246                     listaSobrevivientes.add(player1);
247                     System.out.println(player1.nombre + " HA GANADO LA RONDA -----");
248                 } else {
249                     listaSobrevivientes.add(player2);
250                     System.out.println(player2.nombre+ " HA GANADO LA RONDA -----");
251                 }
252             }
253         }
254     }
255 }
```

Este procedimiento es el de startBattle, que es el encargado de hacer todo lo necesario para El juego y que se defina el ganador, hola inicialmente se crea una lista concursante la cual servirá para ir almacenando a todos los concursantes de la pelea, hoy luego se crea un ciclo guay con el tamaño de la lista de concursantes para poder crear una lista de sobrevivientes que es donde se Irán almacenando todos los concursantes que vayan pasando de ronda, luego se crea un for junto con un nif donde seguirán comparando tanto el daño creado hp por el jugador uno y el jugador 2 hoy se crea hominid y un while donde se verifica si hubo daño o no, haciendo uso de Math max para evitar que haya daño negativo.

Finalmente se verifica la vidaInicial, haciendo una condición donde si la vida inicial del jugador 1 es mayor a 0, este pasa de ronda, de lo contrario, el jugador 2 lo hace.