

Manual de Usuario

THE BATTLE FOR MIDDLE-EARTH



Cristian Alvarez Martinez

Patricio Rojas Condori

Alex Martinez

Índice

Presentacion.....	3
Objetivo del juego:.....	3
Razas disponibles:.....	3
Mecánica de juego:.....	4
Combate:.....	4
Contenido Técnico del Proyecto:.....	5
Clase "MainWindow":.....	5
Clases "Elf , Dwarf i Human":.....	10
Clase "Weapons":.....	12
Clase "WeaponContainer":.....	14
Clase "Fight_Window":.....	14
Clase "Fight y EndFightWindow":.....	23
Clase "FrameWarriors y FrameWeapons":.....	27
Clase "Ranking":.....	30
Clase "WarriorEnemy":.....	35
Clase "BBDD":.....	36
Glosario de términos.....	38
● Clase.....	38
● Métodos.....	38
● Base de datos.....	39
● Interfaz gráfica.....	39
● GitHub.....	39
● Razas.....	39
Errores Comunes.....	39
FAQ.....	40
Webgrafía.....	41
● Stack Overflow (https://stackoverflow.com):.....	41
● GitHub (https://github.com):.....	41
● W3Schools (https://www.w3schools.com):.....	41
● Mozilla Developer Network (https://developer.mozilla.org):.....	41
● Java Documentation (https://docs.oracle.com/javase/):.....	42
● JetBrains (https://www.jetbrains.com):.....	42
● Chat GPT (OpenAI):.....	42

Presentacion

¡Bienvenido al juego *THE BATTLE FOR MIDDLE-EARTH* ! En este juego, te adentrarás en un épico conflicto donde tres razas diferentes, humanos, enanos y elfos, luchan por el alzamiento de su raza en la Tierra Media. Cada raza solo puede utilizar ciertas armas para enfrentarse a sus enemigos. El juego se desarrolla por turnos, en los cuales cada ataque puede tener diferentes resultados que determinarán quién saldrá victorioso.

Batalla de Races es un juego que consiste en simular una batalla uno contra uno entre los diferentes personajes disponibles. Al ejecutar el juego, el usuario puede elegir un guerrero y un arma para enfrentarse a un oponente generado automáticamente por el programa. El objetivo principal es ganar y acumular una puntuación que se refleja en un ranking dentro de la aplicación y en una página web.

Objetivo del juego:

El objetivo principal de *THE BATTLE FOR MIDDLE-EARTH* es liderar a tu raza hacia la victoria, derrotando a tus enemigos y asegurando tu supremacía en la Tierra Media. Escoge a los mejores guerreros de tu raza y equipales con sus mejores armas de combate para asegurar vuestra victoria.

Razas disponibles:

Humanos: Los humanos son valientes y versátiles. Sus armas incluyen espadas, arcos y hachas.

Enanos: Los enanos son resistentes y poderosos. Son expertos en el combate cuerpo a cuerpo y pueden soportar grandes cantidades de daño. Sus armas principales son hachas de combate.

Elfos: Los elfos son ágiles y precisos. Son expertos arqueros y sus armas incluyen arcos, dagas y espadas élficas.

Mecánica de juego:

Pantalla principal: Al iniciar el juego te recibirá una pantalla de inicio que preguntará por el nombre del usuario. Para continuar es importante escribir un nombre y seguidamente darle a “save”. De otra forma el programa no te dejara proseguir con tu aventura.

Selección de guerrero y arma: Una vez escogido nuestro nombre nos tocará decidir por cuál de los personajes decantarnos y, en función de este, el arma que portaremos a la batalla. Una vez escojamos guerrero nuestro rival hará lo mismo de forma aleatoria. Una vez tengamos todo listo podremos darle a “fight” y comenzar con el enfrentamiento.

Turnos: El juego se desarrolla por turnos. Cada jugador tiene la oportunidad de atacar al rival durante su turno, al finalizar su ataque o ataques, el turno pasa al siguiente jugador.

Pantalla final y conclusión: En el momento que uno de los dos guerreros caiga en combate saldrá una pantalla preguntándonos si queremos volver a jugar sin importar si hemos ganado o no. Si escogemos “Si” se nos permitirá volver a pelear y escoger otro personaje si así lo deseamos, en caso contrario el programa finalizará. En cualquier momento tenemos la opción de acceder a una ventana de ranking en la cual podremos ver los usuarios con la mayor puntuación de batalla. Inclusive si salimos invictos de las suficientes batallas podremos ver nuestro ascenso en el ranking en tiempo real.

Combate:

Cuando te enfrentes a un enemigo, se llevará a cabo un combate por turnos. Tendrás la oportunidad de atacar y defenderte de los ataques enemigos. Utiliza estratégicamente tus armas y habilidades para infligir el mayor daño posible a tus oponentes mientras te proteges de sus ataques usando tanto tu defensa como tu agilidad.

Ganar la partida: Para ganar una batalla tienes que debilitar a tu enemigo.

Recuerda que estas son solo las instrucciones básicas para comenzar a jugar "THE BATTLE FOR MIDDLE-EARTH". Asegúrate de explorar todas las combinaciones de guerreros y armas para escoger la que mejor te represente.

Contenido Técnico del Proyecto:

En el desarrollo de "*THE BATTLE FOR MIDDLE-EARTH*", hemos utilizado el lenguaje de programación Java para implementar toda la mecánica del juego. Se han creado diferentes clases, las cuales han sido fundamentales para la creación y funcionamiento del juego.

Puedes encontrar el repositorio de este proyecto en GitHub:

<https://github.com/PatricioGitHub1/Projecte-BatallaRaces>

A continuación, se muestran todas las clases utilizadas:

Clase "MainWindow":

Esta clase es la primera ventana que vemos al iniciar el programa, sirve para recibir al usuario y pedir su nombre. Es desde esta clase que se ha de iniciar el juego.

```
2  import java.awt.BorderLayout;
3
4
5  public class MainWindow extends JFrame {
6      private JPanel panel0, panel1, panel2, panel3, panel4, panel5, panel6, panel7, panel8;
7      private JLabel title, text, creators, error;
8      private JTextField name;
9      private JButton save, start, exit;
10     private String username;
11     private boolean valid = false;
12     private BBDD bd;
13
14
15
16
17
18     public MainWindow() {
19         panel0 = new JPanel();
20         bd = new BBDD();
21         panel1 = new JPanel() {
22             private BufferedImage image;
23             private int newWidth = 1250;
24             private int newHeight = 600;
25
26             @Override
27             protected void paintComponent(Graphics g) {
28                 super.paintComponent(g);
29                 try {
30                     image = ImageIO.read(new File("./imagenes/portada.jpg"));
31                     g.drawImage(image.getScaledInstance(newWidth, newHeight, Image.SCALE_SMOOTH), 0, 0, null);
32                 } catch (IOException e) {
33                     e.printStackTrace();
34                 }
35             }
36         };
37         panel2 = new JPanel();
38         panel3 = new JPanel();
39         panel4 = new JPanel();
40         panel5 = new JPanel();
41         panel6 = new JPanel();
42         panel7 = new JPanel();
43         panel8 = new JPanel();
44         error = new JLabel();
45         title = new JLabel("THE BATTLE FOR MIDDLE-EARTH");
46         title.setFont(new Font("Abyssinica SIL", Font.BOLD, 30));
47         text = new JLabel("Enter your username");
48         creators = new JLabel("By Cristian Alvarez | Alex Martinez | Patricio");
49         creators.setFont(new Font("Abyssinica SIL", Font.ITALIC, 10));
50         name = new JTextField(10);
51         save = new JButton("Save");
52         start = new JButton("Start");
53         exit = new JButton("Exit");
54
55         panel8.add(exit);
56         panel5.add(title);
57         panel2.add(text);
58         panel3.add(name);
59         panel4.add(save);
60         panel4.add(start);
61         panel6.add(creators);
62         panel7.add(error);
63
64
65
66
67
68
69
70
71
72 }
```

```

panel1.setLayout(new BorderLayout());

panel0.add(panel1);
panel0.add(panel5);
panel0.add(panel2, BorderLayout.SOUTH);
panel0.add(panel3);
panel0.add(panel7);
panel0.add(panel14);
panel0.add(panel18);
panel0.add(panel16);

panel0.setLayout(new BoxLayout(panel0, BoxLayout.Y_AXIS));
this.add(panel0);

save.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        username = name.getText();
        //Comprobamos que el username no este en blanco.
        if (username.isBlank()) {
            error.setText("Please, choose and save username");
        }else {
            valid = true;
            error.setText("");
        }
    }
});
start.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

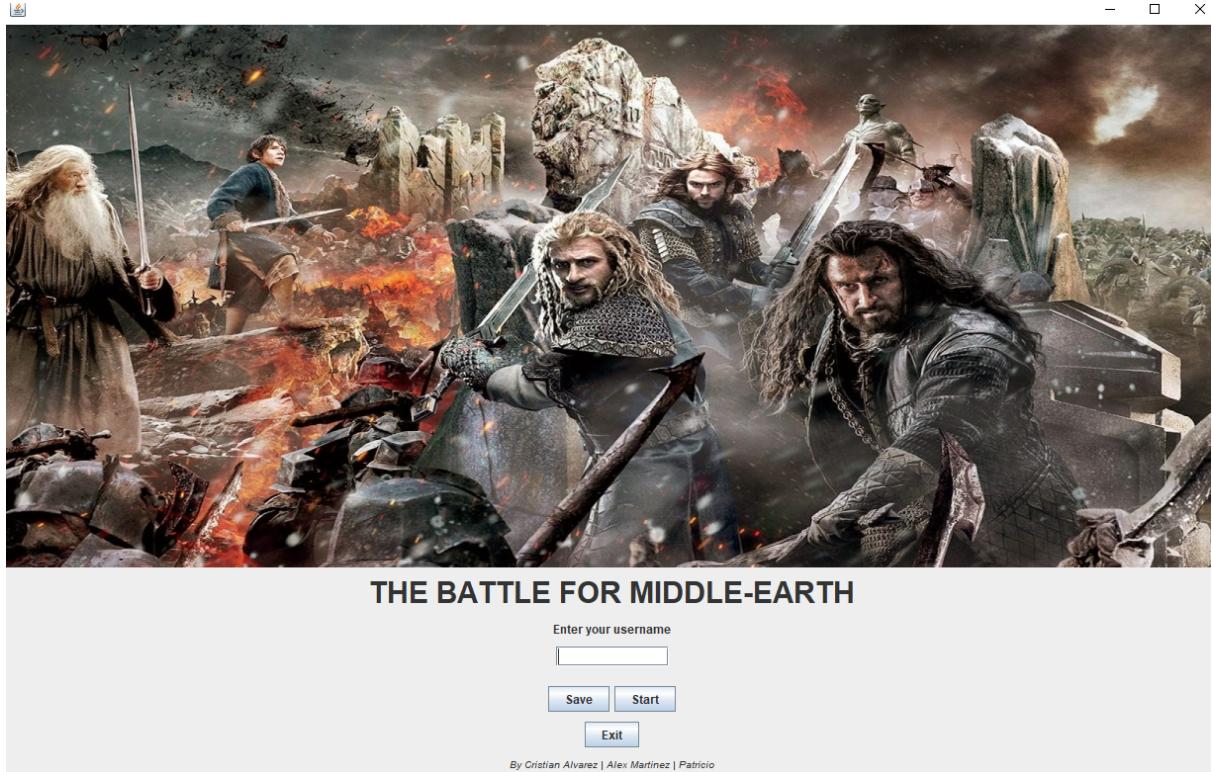
        if (valid == true) {
            //Si valid es true insertamos ese player a la base de datos y creamos una instancia de Fight_Window
            String imagen = "./imagenes/- .jpg";
            bd.insertPlayer(username);
            dispose();
            new Fight_Window(imagen,imagen,imagen,imagen,0,username,null,null,null,null,0);
        }else {
            error.setText("Please, choose and save username");
        }
    }
});

exit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Boton que cierra el programa.
        dispose();
    }
});

setSize(1250, 800);
setResizable(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);
 setLocationRelativeTo(null);
setVisible(true);
}

public static void main(String[] args) {
    new MainWindow();
}

```



Clase "Warrior":

La clase "Warrior" es una clase abstracta que después heredará a las clases "Elf", "Dwarf" y "Human". Estas clases si se pueden instanciar y representan a las tres razas que podemos encontrar en nuestro juego.

```

2 public abstract class Warrior {
3     private int id;
4     private String name;
5     private String image_path;
6     private int health;
7     private int speed;
8     private int agility;
9     private int defense;
0     private int strength;
1     private int initial_health;
2     private int points_value;
3
4@    public Warrior(int id, String name, String image_path) {
5         super();
6         this.id = id;
7         this.name = name;
8         this.image_path = image_path;
9     }
0
1
2
3@    public int getPoints_value() {
4         return points_value;
5     }
6
7
8
9@    public void setPoints_value(int points_value) {
0         this.points_value = points_value;
1     }
2
3
4
5@    public int getId() {
6         return id;
7     }
8
9@    public void setId(int id) {
0         this.id = id;
1     }
2
3@    public String getName() {
4         return name;
5     }
6
7@    public void setName(String name) {
8         this.name = name;
9     }
0
1@    public String getImage_path() {
2         return image_path;
3     }
4
5@    public void setImage_path(String image_path) {
6         this.image_path = image_path;
7     }
8
9@    public int getHealth() {
0         return health;

```

```

public int getHealth() {
    return health;
}

public void setHealth(int health) {
    this.health = health;
}

public int getSpeed() {
    return speed;
}

public void setSpeed(int speed) {
    this.speed = speed;
}

public int getAgility() {
    return agility;
}

public void setAgility(int agility) {
    this.agility = agility;
}

public int getDefense() {
    return defense;
}

public void setDefense(int defense) {
    this.defense = defense;
}

public int getStrength() {
    return strength;
}

public void setStrength(int strength) {
    this.strength = strength;
}

//Metodo para resetear la vida del personaje.
public void resetStats() {
    setHealth(getInitial_health());
}

public int getInitial_health() {
    return initial_health;
}

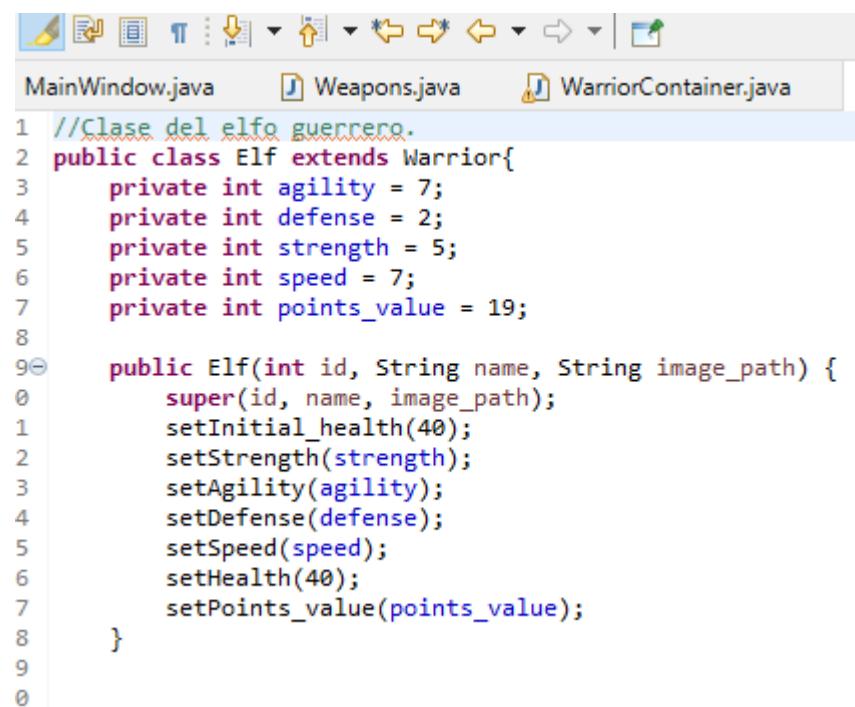
public void setInitial_health(int initial_health) {
    this.initial_health = initial_health;
    this.setHealth(initial_health);
}

```

Clases "Elf , Dwarf i Human":

Són clases que heredan de Warrior y sirven para diferenciar los diferentes tipos de guerreros que hay.

```
1
2 public class Dwarf extends Warrior{
3     private int agility = 5;
4     private int defense = 4;
5     private int strength = 7;
6     private int speed = 3;
7     private int points_value = 21;
8
9     public Dwarf(int id, String name, String image_path) {
10         super(id, name, image_path);
11         setInitial_health(60);
12         setStrength(strength);
13         setAgility(agility);
14         setDefense(defense);
15         setSpeed(speed);
16         setHealth(60);
17         setPoints_value(points_value);
18     }
19
20
21 }
22
```



```
MainWindow.java    Weapons.java    WarriorContainer.java
1 //Clase del elfo guerrero.
2 public class Elf extends Warrior{
3     private int agility = 7;
4     private int defense = 2;
5     private int strength = 5;
6     private int speed = 7;
7     private int points_value = 19;
8
9     public Elf(int id, String name, String image_path) {
0         super(id, name, image_path);
1         setInitial_health(40);
2         setStrength(strength);
3         setAgility(agility);
4         setDefense(defense);
5         setSpeed(speed);
6         setHealth(40);
7         setPoints_value(points_value);
8     }
9
0
```

```

1  public class Human extends Warrior{
2      private int agility = 6;
3      private int defense = 3;
4      private int strength = 7;
5      private int speed = 5;
6      private int points_value = 20;
7
8      public Human(int id, String name, String image_path) {
9          super(id, name, image_path);
10         setInitial_health(50);
11         setStrength(strength);
12         setAgility(agility);
13         setDefense(defense);
14         setSpeed(speed);
15         setHealth(50);
16         setPoints_value(points_value);
17     }
18 }

```

Clase "WarriorContainer":

Esta clase sirve para crear un arraylist con todos los guerreros que hay en la base de datos. Se crearán instancias de las clases anteriores y se añadirán al arraylist.

```

1④ import java.sql.Connection;[]
2
3  public class WarriorContainer {
4      private ArrayList<Warrior> warriorarray;
5      private String urlDatos = "jdbc:mysql://localhost/RacesPAC?serverTimezone=UTC";
6      private String usuario = "root";
7      private String pass = "1234";
8      public ArrayList<Warrior> getWarriorarray() {
9          return warriorarray;
10     }
11     WarriorContainer(){
12         //Inicializamos el arraylist.
13         warriorarray = new ArrayList<Warrior>();
14
15
16
17
18
19
20
21
22
23
24     //Metodo para añadir los guerreros al arraylist.
25     public void addWarrior() {
26         try {
27             //Creamos conexión a la base de datos para obtener todos los datos de cada guerrero.
28             Class.forName("com.mysql.cj.jdbc.Driver");
29             Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
30             String query = "select * from WARRIORS";
31             PreparedStatement ps = conn.prepareStatement(query);
32             ResultSet rs = ps.executeQuery();
33
34             while (rs.next()) {
35                 //Añadimos al arraylist cada guerrero segun el tipo de cada uno.
36                 if (rs.getString(3).substring(11,rs.getString(3).lastIndexOf('.)-1).equals("elfo")) {
37                     warriorarray.add(new Elf(rs.getInt(1),rs.getString(2) , rs.getString(3)));
38                 }else if (rs.getString(3).substring(11,rs.getString(3).lastIndexOf('.)-1).equals("enano")) {
39                     warriorarray.add(new Dwarf(rs.getInt(1),rs.getString(2) , rs.getString(3)));
40                 }else {
41                     warriorarray.add(new Human(rs.getInt(1),rs.getString(2) , rs.getString(3)));
42                 }
43             }
44
45         } catch (ClassNotFoundException e) {
46             System.out.println("EL driver no se a cargado con exito");
47         } catch (SQLException e) {
48             System.out.println("Conexion no creada con exito!!!");
49         }
50     }
51 }
52 }

```

Clase "Weapons":

La clase “Weapons” sirve para representar información de cada una de las armas de la base de datos.

```
public class Weapons {  
    private int id;  
    private String name_weapon;  
    private String weapon_image_path;  
    private int speed;  
    private int strength;  
    private int points_value;  
  
    public Weapons(int id, String name_weapon, String weapon_image_path,int speed,int strength,int points_value) {  
        super();  
        this.id = id;  
        this.name_weapon = name_weapon;  
        this.weapon_image_path = weapon_image_path;  
        this.speed = speed;  
        this.strength = strength;  
        this.points_value = points_value;  
    }  
  
    public int getPoints_value() {  
        return points_value;  
    }  
  
    public void setPoints_value(int points_value) {  
        this.points_value = points_value;  
    }  
  
    public int getSpeed() {  
        return speed;  
    }  
    public void setSpeed(int speed) {  
        this.speed = speed;  
    }  
    public int getStrength() {  
        return strength;  
    }  
}
```

```
}

>     public int getSpeed() {
>         return speed;
>     }

>     public void setSpeed(int speed) {
>         this.speed = speed;
>     }

>     public int getStrength() {
>         return strength;
>     }

>     public void setStrength(int strength) {
>         this.strength = strength;
>     }

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

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

>     public String getName_weapon() {
>         return name_weapon;
>     }

>     public void setName_weapon(String name_weapon) {
>         this.name_weapon = name_weapon;
>     }

>     public String getWeapon_image_path() {
>         return weapon_image_path;
>     }

>     public void setWeapon_image_path(String weapon_image_path) {
>         this.weapon_image_path = weapon_image_path;
>     }

}

}
```

Clase "WeaponContainer":

Al igual que la clase "WarriorContainer", esta sirve para almacenar todas las armas que hay en la base de datos, creando una instancia de la clase Weapons.

```
1④ import java.sql.Connection;⑤
2
3 public class WeaponsContainer {
4     private ArrayList<Weapons> weaponsarray;
5     private String urlDatos = "jdbc:mysql://localhost/RacesPAC?serverTimezone=UTC";
6     private String usuario = "root";
7     private String pass = "1234";
8
9 }
10
11④ public ArrayList<Weapons> getWeaponsarray() {
12     return weaponsarray;
13 }
14
15④ WeaponsContainer() {
16     //inicializamos el arraylist.
17     weaponsarray = new ArrayList<Weapons>();
18 }
19
20 //Metodo para añadir las armas al arraylist.
21④ public void addweapons() {
22     try {
23         Class.forName("com.mysql.cj.jdbc.Driver");
24         Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
25         String query = "select * from WEAPONS";
26         PreparedStatement ps = conn.prepareStatement(query);
27         ResultSet rs = ps.executeQuery();
28
29         while (rs.next()) {
30             weaponsarray.add(new Weapons(rs.getInt(1),rs.getString(2) , rs.getString(3),rs.getInt(4),rs.getInt(5),rs.getInt(6)));
31         }
32
33     } catch (ClassNotFoundException e) {
34         System.out.println("El driver no se a cargado con exito");
35     } catch (SQLException e) {
36         System.out.println("Conexion no creada con exito!!!");
37     }
38 }
39 }
```

Clase "Fight_Window":

La clase "Ventana Lucha" es responsable de mostrar la interfaz gráfica donde se llevan a cabo las batallas. En esta ventana, el jugador puede seleccionar los personajes y las armas que se utilizarán en la batalla. Dependiendo del personaje elegido, se le permitirá al jugador utilizar ciertas armas específicas de su raza.

```
1④ import java.awt.BasicStroke;⑤
2
3 public class Fight_Window extends JFrame {
4     private static Fight_Window instance;
5     private PlayersImage playImage;
6     private JPanel panel0, panel1, panel2, panel3;
7     private JButton chooseCharacter, chooseWeapon, ranking, fight, clear;
8     private JTextArea console;
9     private Fight f;
10    private BDDD bd;
11
12    public Fight_Window(String userimg, String userweapon, String botimg, String botweapon,int id,String username,Warrior warrior_enemy, Weapons weapons_enemy,Warrior user_warrior,Weapons user_weapon,
13    instance = this;
14    panel0 = new JPanel();
15    panel1 = new JPanel();
16    panel2 = new JPanel();
17    panel3 = new JPanel();
18    clear = new JButton("Clear Console");
19    bd = new BDDD();
20
21    console = new JTextArea();
22    console.setEditable(false);
23
24    JScrollPane scrollPane = new JScrollPane(console);
25    clear.addActionListener(new ActionListener() {
26
27        @Override
28        public void actionPerformed(ActionEvent e) {
29            clearConsole();
30        }
31    });
32
33    playimage = new PlayersImage(userimg,userweapon,botimg,botweapon);
34    chooseCharacter = new JButton("Choose Character");
35    chooseWeapon = new JButton("Choose Weapon");
36    ranking = new JButton("Ranking");
37    fight = new JButton("Fight");
38
39    //Estos if son para el tamaño de la barra de la vida
40    if(user_warrior instanceof Elf) {
```

```

77     //Estos if son para el tamaño de la barra de la vida
78     if(user_warrior instanceof Elf) {
79         playimage.userMaximum(40);
80     }else if (user_warrior instanceof Human) {
81         playimage.userMaximum(50);
82     }else if (user_warrior instanceof Dwarf){
83         playimage.userMaximum(60);
84     }
85     if(warrior_enemy instanceof Elf) {
86         playimage.botMaximum(40);
87     }else if (warrior_enemy instanceof Human) {
88         playimage.botMaximum(50);
89     }else if (warrior_enemy instanceof Dwarf){
90         playimage.botMaximum(60);
91     }
92
93
94
95     panel1.add(chooseCharacter);
96     panel1.add(chooseWeapon);
97     panel1.add(ranking);
98     panel2.add(playimage);
99     panel2.add(fight);
100    panel2.add(clear);
101
102    scrollPane.setPreferredSize(new Dimension(200, 80));
103    panel1.setOpaque(false);
104    panel2.setOpaque(false);
105    panel3.setOpaque(false);
106
107    panel0.add(panel1);
108    panel0.add(panel2);
109    panel0.add(panel3);
110    panel0.add(scrollPane);
111    f=new Fight();
112
113    if (user_weapon!=null) {
114        playimage.userAgilitiBar(user_warrior.getAgility()*10);
115        playimage.userDefBar(user_warrior.getDefense()*10);
116        playimage.userPowerBar(user_warrior.getStrength()+user_weapon.getStrength()*10);
117        playimage.userSpeedBar(user_warrior.getSpeed()+user_weapon.getSpeed()*10);
118    }

```

```

116     playimage.userHealthBar(user_warrior.getUserHealth(), 10);
117     playimage.userPowerBar(user_warrior.getStrength() + user_weapon.getStrength() * 10);
118     playimage.userSpeedBar(user_warrior.getSpeed() + user_weapon.getSpeed() * 10);
119     playimage.botAgilityBar(warrior_enemy.getAgility() * 10);
120     playimage.botDefBar(warrior_enemy.getDefense() * 10);
121     playimage.botPowerBar(warrior_enemy.getStrength() + weapons_enemy.getStrength() * 10);
122     playimage.botSpeedBar(warrior_enemy.getSpeed() + weapons_enemy.getSpeed() * 10);
123     }if (user_warrior != null) {
124         playimage.userProgBar(user_warrior.getHealth());
125         playimage.botProgBar(warrior_enemy.getHealth());
126     }
127
128
129
130     ranking.addActionListener(new ActionListener() {
131
132     @Override
133     public void actionPerformed(ActionEvent e) {
134         //Creamos instancia de la clase ranking
135         new Ranking();
136
137     }
138 });
139
140     chooseCharacter.addActionListener(new ActionListener() {
141
142     @Override
143     public void actionPerformed(ActionEvent e) {
144         //Resetea el arraylist de rondas y pone las rondas a cero
145         if (f.getRounds() >= 1) {
146             f.getRoundsarray().clear();
147             user_warrior.resetStats();
148             warrior_enemy.resetStats();
149         }
150
151         dispose();
152         new FrameWarriors(username);
153
154     }
155 });
156
157
chooseWeapon.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (userimg != "/imagenes/- .jpg") {
            clearConsole();
            dispose();
            new FrameWeapons(id, userimg, username, botimg, botweapon, warrior_enemy, weapons_enemy, user_warrior, points);
            user_warrior.resetStats();
            warrior_enemy.resetStats();
            f.getRoundsarray().clear();
        }else {
            consoleText("First you have to choose a Warrior");
        }
    }
});
fight.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //If para comprobar que no se puede pulsar el boton fight sin escoger arma y guerrero.
        if (userimg != "/Imagenes/- .jpg" && userweapon != "/imagenes/- .jpg") {
            if (warrior_enemy.getHealth() <= 0 || user_warrior.getHealth() <= 0) {
                String winner_name = "";
                f.setRounds(0);
                //Vida enemigo o aliado menor o igual que 0, insertamos las rondas y la batalla a la base de datos
                if (warrior_enemy.getHealth() <= 0) {
                    winner_name = username + " has won the Battle!\n";
                    bd.insertBattle(bd.playerId(), user_warrior.getId(), user_weapon.getId(), warrior_enemy.getId(), weapons_enemy.getId(), warrior_enemy.getInitialHealth() - warrior_enemy.getHealth());
                    bd.addRounds(f.getRoundsarray());
                    f.getRoundsarray().clear();
                }else if (user_warrior.getHealth() <= 0) {
                    winner_name = "The Bot has won the Battle!\n";
                    bd.insertBattle(bd.playerId(), user_warrior.getId(), user_weapon.getId(), warrior_enemy.getId(), weapons_enemy.getId(), warrior_enemy.getInitialHealth() - warrior_enemy.getHealth());
                    bd.addRounds(f.getRoundsarray());
                    f.getRoundsarray().clear();
                }
            }
        }
    }
});

```

```

window.java  FrameWeapons.java  Fight.java  Fight.Window.java X
}

    }else if(user_warrior.getHealth()<=0) {
        winner_name = "The Bot has won the Battle!\n";
        bd.insertBattle(bd.playerId(),user_warrior.getId(),user_weapon.getId(),warrior_enemy.getId(),weapons_enemy.getId(),warrior_enemy.getInitial_health()-warrior_enemy.getHealth());
        bd.addRounds(f.getRoundsarray());
        f.getRoundsarray().clear();
    }
    //Creamos instancia para la ventana JOptionPane, cuando terminas una batalla, donde te pide si quieres continuar luchando.
    new EndFightWindow(user_warrior,warrior_enemy,username,f.getTotal_points(),user_weapon,winner_name);
}else {
    //Llamamos al metodo de la lucha, para realizar el combate
    f.singleFight(username, warrior_enemy, weapons_enemy, user_warrior, user_weapon,points);
    //Llamamos a los metodos para cambiar la salud de las barras y los atributos de cada personaje.
    playImage.userProgBar(user_warrior.getHealth());
    playImage.botProgBar(warrior_enemy.getHealth());
}
}

});
```

the.add(panel0);
panel0.setLayout(new BoxLayout(panel0, BoxLayout.Y_AXIS));
setSize(1250, 800);
setResizable(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setLocationRelativeTo(null);
setVisible(true);

```

public void clearConsole() {
    console.setText("");
}
public void consoleText(String text) {
    console.append(text+"\n");
}
public static Fight_Window getInstance() {
    return instance;
}
```

```

class PlayersImage extends JPanel{
    private JPanel panel2,panel3;

    private BufferedImage userImage;
    private BufferedImage botImage,versus;
    private int characterimageWidth = 501;
    private int characterimageHeight = 370;
    private JLabel usupower,usugility,ususpeed,usudfense,botpower,botagility,botspeed,botdfense;

    private BufferedImage userweaponImage;
    private BufferedImage botweaponImage;
    private int weaponimageWidth = 140;
    private int weaponimageHeight = 98;

    private JProgressBar barbot,baruser,barusupower,barusuagility,barusuuspeed,barusudfense,barbotpower,barbotagility,barbotspeed,barbotdfense;
    private String userimg1,userweapon1,botimg1,botweapon1;
```

```

public PlayersImage(String userimg, String userweapon, String botimg, String botweapon) {
    userimg1 = userimg;
    userweapon1 = userweapon;
    botimg1 = botimg;
    botweapon1 = botweapon;
    setPreferredSize(new Dimension(1250,525));
    setOpaque(false);

    panel2 = new JPanel();
    panel3 = new JPanel();
    panel2.setLayout(new BoxLayout(panel2, BoxLayout.Y_AXIS));
    panel3.setLayout(new BoxLayout(panel3, BoxLayout.Y_AXIS));

    usupower = new JLabel("Power");
    usugility = new JLabel("Agility");
    ususpeed = new JLabel("Speed");
    usudfense = new JLabel("Defense");

    usupower.setFont(new Font("Arial", Font.BOLD, 20));
    usugility.setFont(new Font("Arial", Font.BOLD, 20));
    ususpeed.setFont(new Font("Arial", Font.BOLD, 20));
```

```

33 usupower.setFont(new Font("Arial", Font.BOLD, 20));
34 usuagility.setFont(new Font("Arial", Font.BOLD, 20));
35 ususpeed.setFont(new Font("Arial", Font.BOLD, 20));
36 usudefense.setFont(new Font("Arial", Font.BOLD, 20));
37
38 botpower = new JLabel("Power");
39 botagility = new JLabel("Agility");
40 botspeed = new JLabel("Speed");
41 botdefense = new JLabel("Defense");
42
43 botpower.setFont(new Font("Arial", Font.BOLD, 20));
44 botagility.setFont(new Font("Arial", Font.BOLD, 20));
45 botspeed.setFont(new Font("Arial", Font.BOLD, 20));
46 botdefense.setFont(new Font("Arial", Font.BOLD, 20));
47 Font font = new Font("Arial", Font.BOLD, 15);
48
49 // BARRA PROGRESION ARMA
50 barusupower = new JProgressBar();
51 barusuagility = new JProgressBar();
52 barususpeed = new JProgressBar();
53 barusudefense = new JProgressBar();
54
55 barbotpower = new JProgressBar();
56 barbotagility = new JProgressBar();
57 barbotspeed = new JProgressBar();
58 barbotdefense = new JProgressBar();
59
60 barusupower.setMinimum(0);
61 barusupower.setMaximum(100);
62 barusupower.setValue(0);
63 barusupower.setForeground(Color.red);
64 barusupower.setBackground(Color.white);
65
66 barusuagility.setMinimum(0);
67 barusuagility.setMaximum(100);
68 barusuagility.setValue(0);
69 barusuagility.setForeground(Color.green);
70 barusuagility.setBackground(Color.white);
71
72 barususpeed.setMinimum(0);
73 barususpeed.setMaximum(100);
74 barususpeed.setValue(0);

```

```
barususpeed.setForeground(Color.blue);
barususpeed.setBackground(Color.white);

barusudefense.setMinimum(0);
barusudefense.setMaximum(100);
barusudefense.setValue(0);
barusudefense.setForeground(Color.darkGray);
barusudefense.setBackground(Color.white);
//-----
barbotpower.setMinimum(0);
barbotpower.setMaximum(100);
barbotpower.setValue(0);
barbotpower.setForeground(Color.red);
barbotpower.setBackground(Color.white);

barbotagility.setMinimum(0);
barbotagility.setMaximum(100);
barbotagility.setValue(0);
barbotagility.setForeground(Color.green);
barbotagility.setBackground(Color.white);

barbotspeed.setMinimum(0);
barbotspeed.setMaximum(100);
barbotspeed.setValue(0);
barbotspeed.setForeground(Color.blue);
barbotspeed.setBackground(Color.white);

barbotdefense.setMinimum(0);
barbotdefense.setMaximum(100);
barbotdefense.setValue(0);
barbotdefense.setForeground(Color.darkGray);
barbotdefense.setBackground(Color.white);

panel2.add(usupower);
panel2.add(usuagility);
panel2.add(ususpeed);
panel2.add(usudefense);

panel3.add(botpower);
panel3.add(botagility);
panel3.add(botspeed);
panel3.add(botdefense);
```



```
this.add(barusupower);
this.add(barusuagility);
this.add(barususpeed);
this.add(barusudefense);

this.add(barbotpower);
this.add(barbotagility);
this.add(barbotspeed);
this.add(barbotdefense);

barusupower.setBounds(300, 431, 201, 22);
barusuagility.setBounds(300, 453, 201, 22);
barususpeed.setBounds(300, 474, 201, 22);
barusudefense.setBounds(300, 496, 201, 22);

barbotpower.setBounds(1049, 431, 201, 22);
barbotagility.setBounds(1049, 453, 201, 22);
barbotspeed.setBounds(1049, 474, 201, 22);
barbotdefense.setBounds(1049, 496, 201, 22);

// BARRA PROGRESION USUARIOS
barbot = new JProgressBar();
barbot.setMinimum(0);
barbot.setMaximum(0);
barbot.setStringPainted(true);
barbot.setValue(0);
barbot.setForeground(Color.green);
barbot.setBackground(Color.RED);
barbot.setFont(font);

baruser = new JProgressBar();
baruser.setMinimum(0);
baruser.setMaximum(0);
baruser.setStringPainted(true);
baruser.setValue(0);
baruser.setStringPainted(true);
baruser.setValue(0);
baruser.setForeground(Color.green);
```

<

```

12     this.add(barbot);
13     this.add(baruser);
14     baruser.setBounds(0, 1, 501, 60);
15     barbot.setBounds(750, 1, 500, 60);
16     this.setLayout(null);
17     panel2.setBounds(150, 431, 100, 95);
18     panel3.setBounds(900, 431, 100, 95);
19     this.add(panel2);
20     this.add(panel3);
21
22
23 }
24
25 @Override
26 protected void paintComponent(Graphics g) {
27     super.paintComponent(g);
28     // Link de la foto character
29     String linkcharacteruser = userimg1;
30     String linkcharacterbot = botimg1;
31     // Link de la foto weapon
32     String linkweaponuser = userweapon1;
33     String linkweaponbot = botweapon1;
34     try {
35         //foto usuario
36         versus = ImageIO.read(new File("./imagenes/versus.jpg"));
37         g.drawImage(versus.getScaledInstance(characterimageWidth, characterimageHeight, Image.SCALE_SMOOTH), 385, 61, null);
38     } catch (IOException e) {
39         e.printStackTrace();
40     }
41     // IMAGENES CHARACTERS
42     try {
43         //foto usuario
44         userImage = ImageIO.read(new File(linkcharacteruser));
45         g.drawImage(userImage.getScaledInstance(characterimageWidth, characterimageHeight, Image.SCALE_SMOOTH), 0, 61, null);
46     } catch (IOException e) {
47         e.printStackTrace();
48     }
49     try {
50         //foto bot
51         botImage = ImageIO.read(new File(linkcharacterbot));
52         g.drawImage(botImage.getScaledInstance(characterimageWidth, characterimageHeight, Image.SCALE_SMOOTH), 750, 61, null);
53     }

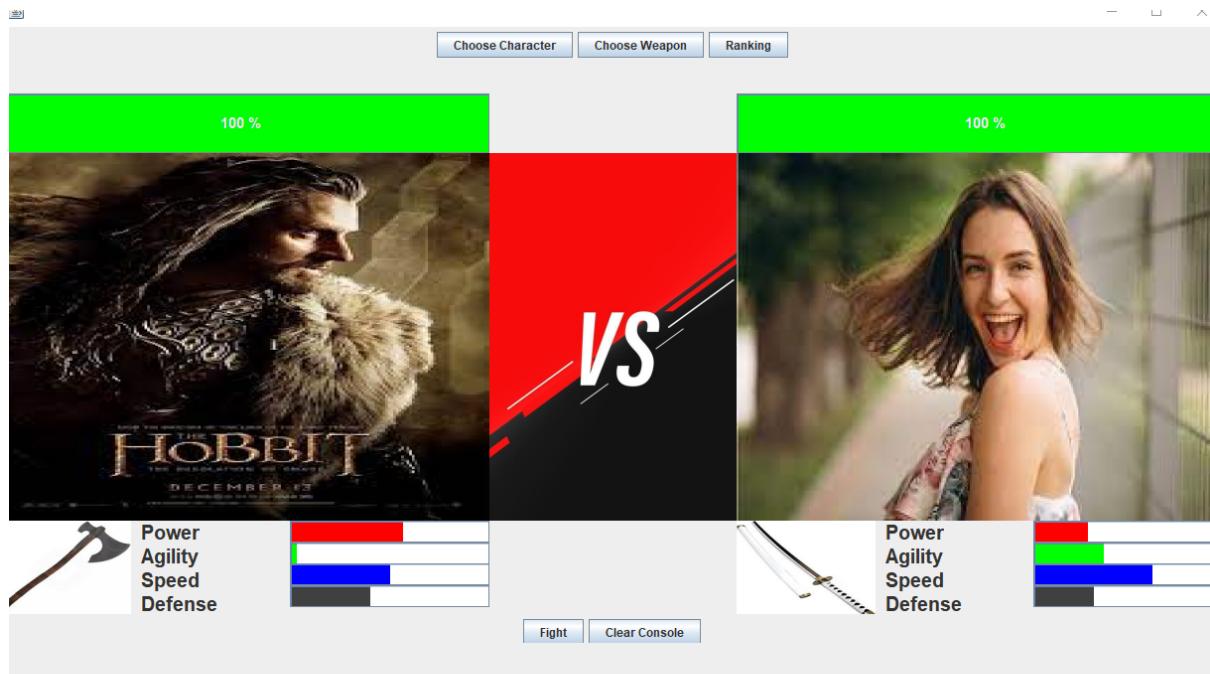
```

```

54     //IMAGENES WEAPONS
55     try {
56         //foto usuario
57         userweaponImage = ImageIO.read(new File(linkweaponuser));
58         g.drawImage(userweaponImage.getScaledInstance(weaponimageWidth, weaponimageHeight, Image.SCALE_SMOOTH), 0, 431, null);
59     } catch (IOException e) {
60         e.printStackTrace();
61     }
62     try {
63         //foto bot
64         botweaponImage = ImageIO.read(new File(linkweaponbot));
65         g.drawImage(botweaponImage.getScaledInstance(weaponimageWidth, weaponimageHeight, Image.SCALE_SMOOTH), 750, 431, null);
66     } catch (IOException e) {
67         e.printStackTrace();
68     }
69 }
70
71 /*-----*/
72 public void userProgBar(Integer total) {
73     baruser.setValue(total);
74 }
75
76 public void botProgBar(Integer total) {
77     barbot.setValue(total);
78 }
79
80 public void userMaximum(Integer total) {
81     baruser.setMaximum(total);
82 }
83
84 public void botMaximum(Integer total) {
85     barbot.setMaximum(total);
86 }
87
88 /*-----*/
89 public void userPowerBar(Integer total) {
90     baruserpower.setValue(total);
91 }
92

```

```
77     /*-----*/
78     public void userProgBar(Integer total) {
79         baruser.setValue(total);
80     }
81     public void botProgBar(Integer total) {
82         barbot.setValue(total);
83     }
84     public void userMaximum(Integer total) {
85         baruser.setMaximum(total);
86     }
87     public void botMaximum(Integer total) {
88         barbot.setMaximum(total);
89     }
90     /*-----*/
91     public void userPowerBar(Integer total) {
92         barusupower.setValue(total);
93     }
94     public void userSpeedBar(Integer total) {
95         barusuagility.setValue(total);
96     }
97     public void userAgilitiBar(Integer total) {
98         barususpeed.setValue(total);
99     }
100    public void userDefBar(Integer total) {
101        barusudefense.setValue(total);
102    }
103    /*-----*/
104    public void botPowerBar(Integer total) {
105        barbotpower.setValue(total);
106    }
107    public void botSpeedBar(Integer total) {
108        barbotagility.setValue(total);
109    }
110    public void botAgilitiBar(Integer total) {
111        barbotspeed.setValue(total);
112    }
113    public void botDefBar(Integer total) {
114        barbotdefense.setValue(total);
115    }
116
117 }
118 <
```



Clase "Fight y EndFightWindow":

La clase "Fight" contiene toda la lógica y el mecanismo necesario para llevar a cabo la batalla entre los personajes. Aquí se implementan las reglas de combate, se calculan los puntos de daño, se aplican los efectos de las armas y se determina el resultado de la batalla. Es en esta clase donde se gestionan los turnos y las acciones de los personajes durante el combate.

"EndFightWindow" es una clase que sirve para gestionar si el usuario quiere volver a jugar otra batalla o no. Se muestra una ventana final con la opción de "Si" o "No".

```

1@ import java.util.ArrayList;
4
5 public class Fight{
6     private int enemy_health;
7     private int player_health;
8     private int total_points=0;
9     private boolean turn;
10    private int rounds;
11    private ArrayList<Rounds> roundsarray;
12    private int totalDamage;
13    private BBDD bd;
14
15@     public ArrayList<Rounds> getRoundsarray() {
16         return roundsarray;
17     }
18
19@     public int getTotalDamage() {
20         return totalDamage;
21     }
22
23@     public int getRounds() {
24         return rounds;
25     }
26
27@     public void setRounds(int rounds) {
28         this.rounds = rounds;
29     }
30
31
32
33@     public Fight() {
34         roundsarray = new ArrayList<Rounds>();
35         bd = new BBDD();
36     }
37
38@     public void singlefight (String username,Warrior warrior_enemy, Weapons weapons_enemy,Warrior user_warrior,Weapons user_weapon, int points) {
39         Fight_Window vn = Fight_Window.getInstance();
40         rounds++;
41         if (rounds == 1) {
42             turn = getFasterWarrior(user_warrior, warrior_enemy, user_weapon, weapons_enemy);
43             total_points=warrior_enemy.getPoints_value()+weapons_enemy.getPoints_value()+points;
44         }
45     }
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
0

```

Console X

```

minated> MainWindow [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (17 may 2023 20:54:29 - 21:00:14) [nid: 19500]

```

```

9     Fight_Window vn = Fight_Window.getInstance();
0     rounds++;
1     if (rounds == 1) {
2         turn = getFasterWarrior(user_warrior, warrior_enemy, user_weapon, weapons_enemy);
3         total_points=warrior_enemy.getPoints_value()+weapons_enemy.getPoints_value()+points;
4     }
5
6     //FALSO = BOT, VERDADERO = HUMANO
7     int percentageAtkHits = 1 + (int) (Math.random() * 101);
8     vn.consoleText("=====");
9     if (turn) {
0         vn.consoleText("Round number "+rounds+" | Turn of "+username);
1         turn = roundFight(user_warrior, warrior_enemy,user_weapon,weapons_enemy,turn);
2         roundsarray.add(new Rounds(rounds,0,user_warrior.getId(),user_weapon.getId(),warrior_enemy.getId(),weapons_enemy.getId(),getTotalDamage(),0));
3     } else if (!turn) {
4         vn.consoleText("Round number "+rounds+" | Turn of Bot");
5         turn = roundFight(warrior_enemy, user_warrior,weapons_enemy,user_weapon,turn);
6         roundsarray.add(new Rounds(rounds,0,user_warrior.getId(),user_weapon.getId(),warrior_enemy.getId(),weapons_enemy.getId(),0,getTotalDamage()));
7     }
8
9
0
1
2
3
4
5
6
7@     boolean roundFight(Warrior attacker, Warrior defender,Weapons weapon_attacker,Weapons weapon_defender,boolean turno) {
8         Fight_Window vn = Fight_Window.getInstance();
9         vn.consoleText("=====");
0         int percentageAtkHits = 1 + (int) (Math.random() * 101);
1         if ((attacker.getAgility()) * 10 > percentageAtkHits) {
2             vn.consoleText("The attack was succesful");
3             int percentageDodge = 1 + (int) (Math.random() * 51);
4
5             if ((defender.getAgility()) > percentageDodge) {
6                 vn.consoleText(defender.getName()+" has dodged the attack");
7                 vn.consoleText("Role swap");
8                 return changeTurn(turno);
9             } else {
0                 //EN ESTE CASO EL ATAQUE HA TENIDO EXITO Y EL DEFENSOR NO HA PODIDO DEFENDERSE
1
2
3
4
5
6
7
8
9
0

```

```

78     return changeTurn(turno);
79 }
80 } else {
81     //EN ESTE CASO EL ATAQUE HA TENIDO EXITO Y EL DEFENSOR NO HA PODIDO DEFENDERSE
82     totalDamage = attacker.getStrength() + weapon_attacker.getStrength() - defender.getDefense();
83     int updatedDefHealth = defender.getHealth() - totalDamage;
84     defender.setHealth(updatedDefHealth);
85     vn.consoleText(defender.getName() + " has received "+totalDamage+ " damage points from "+attacker.getName());
86     //AQUI MODIFICAR LA BARRITA DE SALUD DEL DEFENSOR
87     //
88     //
89     //Ver si atacante puede repetir turno
90     if (attacker.getSpeed() + weapon_attacker.getSpeed() <= defender.getSpeed()+weapon_defender.getSpeed()) {
91         vn.consoleText("Role swap");
92         return changeTurn(turno);
93     }
94 }
95
96 //En el caso que la velocidad del defensor no sea mayor al atacante
97 int changeToSwap = 1 + (int) (Math.random() * 101);
98 if ((attacker.getSpeed() + weapon_attacker.getSpeed() - defender.getSpeed() - weapon_defender.getSpeed())*10 > changeToSwap) {
99     vn.consoleText("The attacker keeps the offensive");
100    return turno;
101 }
102 vn.consoleText("Role swap");
103 return changeTurn(turno);
104 }
105
106 } else {
107     vn.consoleText("The attack failed");
108     vn.consoleText("Role swap");
109     return changeTurn(turno);
110 }
111
112 boolean changeTurn(boolean current) {
113     if (current) {
114         return false;
115     } else {
116         return true;
117     }
118 }
119 }

14
15     if (current) {
16         return false;
17     } else {
18         return true;
19     }
20 }

21 boolean getFasterWarrior(Warrior user,Warrior bot,Weapons user1, Weapons bot1) {
22     //True es humano y False es el bot
23     if (user.getSpeed()+user1.getSpeed() == bot.getSpeed()+bot1.getSpeed()) {
24         if (user.getAgility() == bot.getAgility()) {
25             int random_index = (int) Math.random()*2;
26             if (random_index == 0) {
27                 return false;
28             } else {
29                 return true;
30             }
31         } else if (user.getAgility() > bot.getAgility()) {
32             return true;
33         } else {
34             return false;
35         }
36     } else if (user.getSpeed()+user1.getSpeed() > bot.getSpeed()+bot1.getSpeed()) {
37         return true;
38     } else {
39         return false;
40     }
41 }

43

44 public int getEnemy_health() {
45     return enemy_health;
46 }
47
48

49 public void setEnemy_health(int enemy_health) {
50     this.enemy_health = enemy_health;
51 }
52
53
54 public int getPlayer_health() {
55     return player_health;
56 }

```

```

0     this.enemy_health = enemy_health;
1 }
2
3
4@ public int getPlayer_health() {
5     return player_health;
6 }
7
8
9@ public void setPlayer_health(int player_health) {
0     this.player_health = player_health;
1 }
2
3
4@ public int getTotal_points() {
5     return total_points;
6 }
7
8
9@ public void setTotal_points(int total_points) {
0     this.total_points = total_points;
1 }
2
3
4@ public static void main(String[] args) {
5
6 }
7
8 }
9
0 class EndFightWindow {
1
2@ public EndFightWindow(Warrior user,Warrior bot,String username,int total_points,Weapons weapon, String winner_name) {
3     String message = winner_name+"Do you want to keep fighting?";
4     int answer;
5     answer = JOptionPane.showOptionDialog(null, message, "End Of Battle", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);
6     actionJOptionPane(answer, user,bot,username,total_points,weapon);
7 }
8
9@ public void actionJOptionPane(int single_case, Warrior user, Warrior bot, String username,int total_points,Weapons weapon) {
0     Fight_Window vn = Fight_Window.getInstance();
1     BBDD bd = new BBDD();
2
3
4@ class EndFightWindow {
1
2@ public EndFightWindow(Warrior user,Warrior bot,String username,int total_points,Weapons weapon, String winner_name) {
3     String message = winner_name+"Do you want to keep fighting?";
4     int answer;
5     answer = JOptionPane.showOptionDialog(null, message, "End Of Battle", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);
6     actionJOptionPane(answer, user,bot,username,total_points,weapon);
7 }
8
9@ public void actionJOptionPane(int single_case, Warrior user, Warrior bot, String username,int total_points,Weapons weapon) {
0     Fight_Window vn = Fight_Window.getInstance();
1     BBDD bd = new BBDD();
2     if (single_case == JOptionPane.YES_OPTION) {
3         //CASE ANSWER IS YES, AND CHECK IF USER IS ALIVE
4         if (user.getHealth() <= 0) {
5             //VIDA MENOR QUE 0
6             user.resetStats();
7             bot.resetStats();
8             String imagen="./imagenes/-1.jpg";
9             vn.dispose();
10            new Fight_Window(imagen,imagen,imagen,imagen,0,username,null,null,null,0);
11        } else {
12            //VIDA MAYOR QUE 0
13            user.resetStats();
14            bot.resetStats();
15            bd.updatePlayer(username, total_points);
16            WarriorEnemy we = new WarriorEnemy();
17            we.Enemy_Random();
18            vn.dispose();
19            new Fight_Window(user.getImage_path(),weapon.getWeapon_image_path() , we.getWarrior_enemy().getImage_path(), we.getWeapon().getWeapon_image_path(), user.getId(), username, we.getWarrior()
20        }
21    } else {
22        //CASE ANSWER IS NO,//
23        vn.dispose();
24        bd.updatePlayer(username, total_points);
25    }
26 }
27 }
28 }
29 }

```

Clase "FrameWarriors y FrameWeapons":

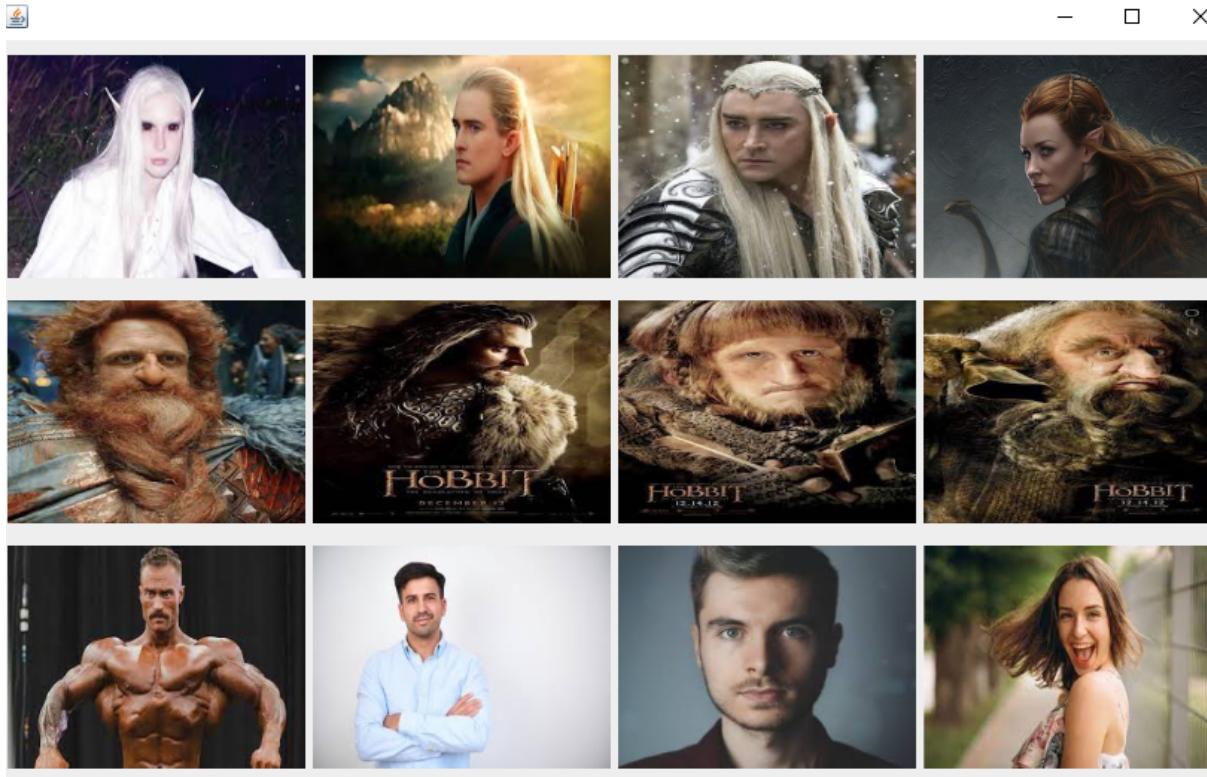
Estas clases muestran una interfaz gráfica con la que podemos elegir un personaje y un arma.

The screenshot shows a Java IDE interface with two tabs: 'FrameWarriors.java' and 'Console'. The code in 'FrameWarriors.java' is as follows:

```
1① import java.awt.Dimension;
2
3 public class FrameWarriors extends JFrame{
4
5     private JPanel p0,p1,p2,p3;
6     private ArrayList<JLabel> arraylabels;
7     private String userwarriorpath,botwarriorpath,botweaponpath;
8     private Warrior userwarrior;
9
10
11     FrameWarriors(String username){
12         arraylabels = new ArrayList<JLabel>();
13         WarriorContainer wl = new WarriorContainer();
14         wl.addWArrior();
15         //For para crear labels con cada imagen de los guerreros.
16         for (int i=0;i<wl.getWarriorarray().size();i++) {
17             ImageIcon icono = new ImageIcon(wl.getWarriorarray().get(i).getImage_path());
18             Image imagenOriginal = icono.getImage();
19             Image imagenRedimensionada = imagenOriginal.getScaledInstance(200, 150, Image.SCALE_SMOOTH);
20             ImageIcon iconoRedimensionado = new ImageIcon(imagenRedimensionada);
21             JLabel label = new JLabel(iconoRedimensionado);
22             arraylabels.add(label);
23         }
24         //For para crear Mouse Listener en cada imagen, sirve para que cuando cliquemos en una imagen elija a ese personaje
25         for (int i = 0; i<arraylabels.size();i++) {
26             JLabel label = arraylabels.get(i);
27             arraylabels.get(i).addMouseListener(new MouseAdapter() {
28
29                 public void mouseClicked(MouseEvent e) {
30                     int index = arraylabels.indexOf(label);
31
32                     dispose();
33                     //Creamos instancia del enemigo, que se elige aleatoriamente y se lo pasamos al constructor de Fight_Window.
34                     WarriorEnemy we = new WarriorEnemy();
35                     we.Enemy_Random();
36                     Warrior warrior_enemy = we.getWarrior_enemy();
37                     Weapons weapon_enemy = we.getWeapon();
38                     userwarriorpath=wl.getWarriorarray().get(index).getImage_path();
39                     botwarriorpath = we.getWarrior_enemy().getImage_path();
40                     botweaponpath = we.getWeapon().getImage_path();
41                     botweaponpath = we.getWeapon().getImage_path();
42
43                     new Fight_Window(userwarriorpath, "./imagenes/- .jpg",botwarriorpath ,botweaponpath,id_warrior,username,warrior_enemy,weapon_enemy,userwarrior,null,0);
44
45                 }
46             });
47         });
48     }
49
50     p0 = new JPanel();
51     p1 = new JPanel();
52     p2 = new JPanel();
53     p3 = new JPanel();
54     for (int i = 0;i<4;i++) {
55         p1.add(arraylabels.get(i));
56     }
57     for (int i = 4;i<8;i++) {
58         p2.add(arraylabels.get(i));
59     }
60     for (int i = 8;i<12;i++) {
61         p3.add(arraylabels.get(i));
62     }
63
64     p0.add(p1);
65     p0.add(p2);
66     p0.add(p3);
67     add(p0);
68     setSize(850,550);
69     setDefaultCloseOperation(EXIT_ON_CLOSE);
70     setLocationRelativeTo(null);
71     setVisible(true);
72 }
```

The 'Console' tab shows the output of the application:

```
minated> MainWindow [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (17 may 2023, 20:41:12 – 20:41:24) [pid: 19376]
```



```

14 import java.awt.BorderLayout;|
15
16 public class FrameWeapons extends JFrame{
17
18     private JPanel p0,p1;
19     private ArrayList<JLabel> arraylabels;
20     private String urlDatos = "jdbc:mysql://localhost/RacesPAC?serverTimezone=UTC";
21     private String usuario = "root";
22     private String pass = "1234";
23     private String userweaponpath;
24     private Weapons userweapon;
25
26     FrameWeapons(int id,String warriorpath,String username,String botimg, String botweapon,Warrior warrior_enemy, Weapons weapons_enemy,Warrior user_warrior,int points){
27         ArrayList<Integer> arrayid = new ArrayList<Integer>();
28         arraylabels = new ArrayList<JLabel>();
29         WeaponsContainer w1 = new WeaponsContainer();
30         w1.addWeapons();
31         try {
32             //Creamos una conexión a la base de datos para sacar las armas disponibles para el guerrero seleccionado, para eso utilizamos la tabla de WEAPONS_AVAILABLE.
33             Class.forName("com.mysql.cj.jdbc.Driver");
34             Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
35             String query = "select Weapon_id from WEAPONS_AVAILABLE where Warrior_id="+id;
36             PreparedStatement ps = conn.prepareStatement(query);
37             ResultSet rs = ps.executeQuery();
38             while(rs.next()) {
39                 arrayid.add(rs.getInt(1));
40             }
41             //For que sirve para crear labels que contienen las imágenes de las armas disponibles.
42             for (int i=0;i<w1.getWeaponsarray().size();i++) {
43                 for (int x = 0;x<arrayid.size() ;x++) {
44                     if (w1.getWeaponsarray().get(i).getId()==arrayid.get(x)) {
45                         ImageIcon icono = new ImageIcon(w1.getWeaponsarray().get(i).getWeapon_image_path());
46                         Image imagenOriginal = icono.getImage();
47                         Image imagenRedimensionada = imagenOriginal.getScaledInstance(200, 150, Image.SCALE_SMOOTH);
48                         ImageIcon iconoRedimensionado = new ImageIcon(imagenRedimensionada);
49                         JLabel label = new JLabel(iconoRedimensionado);
50                         arraylabels.add(label);
51                     }
52                 }
53             }
54         }
55     }
56     //For para añadir MouseListener a las labels, para que al pulsar esa imagen se guarde el arma.
57     for (int i = 0; i<arraylabels.size();i++) {
58

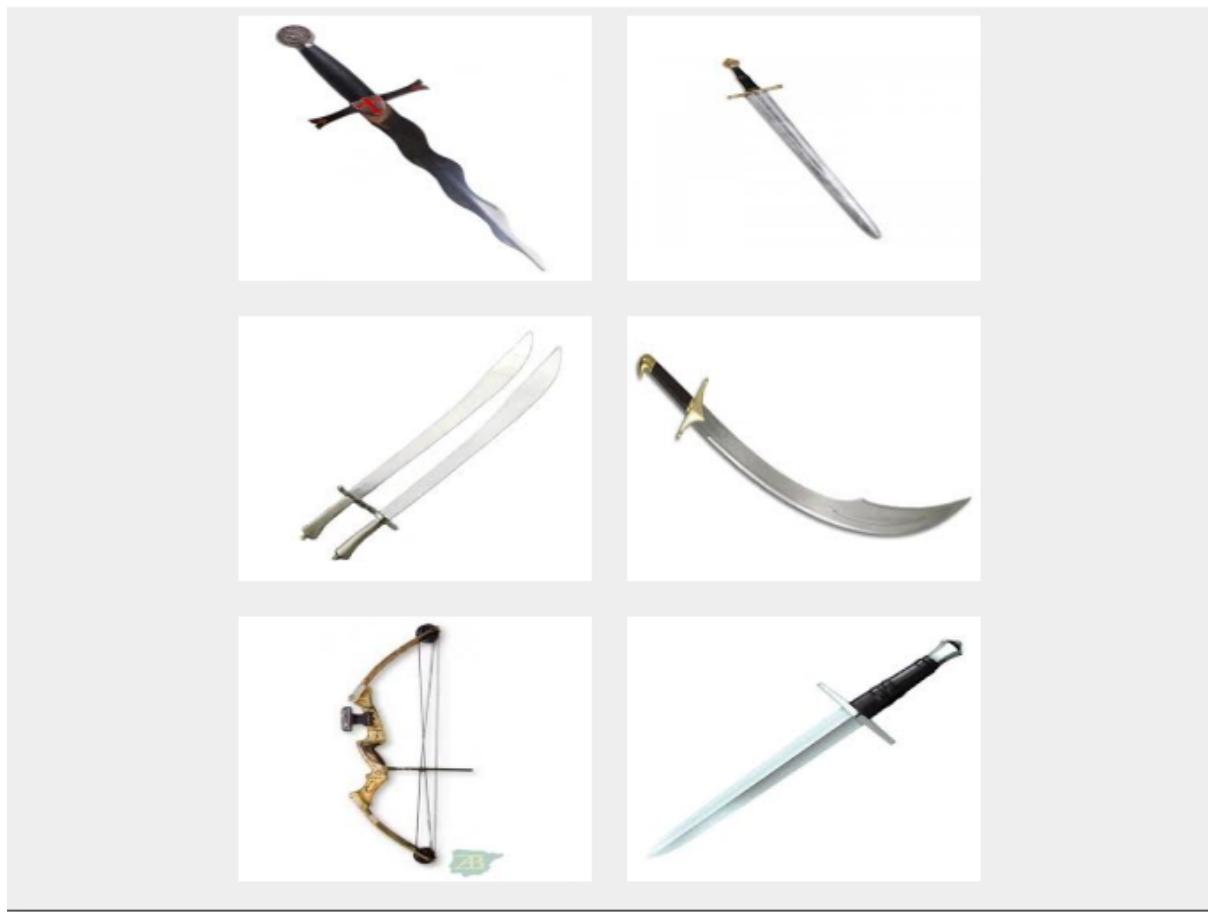
```

```

16
17     }
18     //For para añadir MouseListener a las labels, para que al pulsar esa imagen se guarde el arma.
19     for (int i = 0; i<arraylabels.size();i++) {
20         JLabel label = arraylabels.get(i);
21         arraylabels.get(i).addMouseListener(new MouseAdapter() {
22
22             public void mouseClicked(MouseEvent e) {
23                 int z = 0;
24                 int index = arraylabels.indexOf(label);
25                 for (int x = 0;<wl.getWeaponsarray().size();x++) {
26                     if (wl.getWeaponsarray().get(x).getId()==arrayid.get(index)) {
27                         z = x;
28                         break;
29                     }
30                 }
31                 dispose();
32                 //Guardamos el path y el arma.
33                 userweaponpath = wl.getWeaponsarray().get(z).getWeapon_image_path();
34                 userweapon = wl.getWeaponsarray().get(z);
35                 new Fight_Window(warriorthpath,userweaponpath,botimg,botweapon,id,username,warrior_enemy,weapons_enemy,user_warrior,userweapon,points);
36             }
37         });
38     }
39
40     p0 = new JPanel();
41     p1 = new JPanel();
42     p1.setLayout(new GridLayout(3,3,20,20));
43
44     for (int i = 0;i<arraylabels.size();i++) {
45         p1.add(arraylabels.get(i));
46     }
47
48     p0.add(p1);
49     add(p0);
50     setSize(700,550);
51     setDefaultCloseOperation(EXIT_ON_CLOSE);
52     setLocationRelativeTo(null);
53     setVisible(true);
54 } catch (ClassNotFoundException e) {
55
56     }
57     dispose();
58     //Guardamos el path y el arma.
59     userweaponpath = wl.getWeaponsarray().get(z).getWeapon_image_path();
60     userweapon = wl.getWeaponsarray().get(z);
61     new Fight_Window(warriorthpath,userweaponpath,botimg,botweapon,id,username,warrior_enemy,weapons_enemy,user_warrior,userweapon,points);
62 }
63
64     p0 = new JPanel();
65     p1 = new JPanel();
66     p1.setLayout(new GridLayout(3,3,20,20));
67
68     for (int i = 0;i<arraylabels.size();i++) {
69         p1.add(arraylabels.get(i));
70     }
71
72     p0.add(p1);
73     add(p0);
74     setSize(700,550);
75     setDefaultCloseOperation(EXIT_ON_CLOSE);
76     setLocationRelativeTo(null);
77     setVisible(true);
78 } catch (ClassNotFoundException e) {
79     System.out.println("EL driver no se a cargado con exito");
80 } catch (SQLException e) {
81     System.out.println("Conexion no creada con exito!!!");
82 }
83 }
```



- □ ×



Clase "Ranking":

La clase ranking muestra una interfaz gráfica con un ranking de los 10 jugadores con más puntos ordenados descendenteamente.

```

2
3④ import java.awt.BasicStroke;□
44
45 public class Ranking extends JFrame {
46     private JPanel p0,p1,p2,p3,panel1,p4;
47     private JLabel subtítulo,creators;
48     private JButton exit;
49     private String urlDatos = "jdbc:mysql://localhost/RacesPAC?serverTimezone=UTC";
50     private String usuario = "root";
51     private String pass = "1234";
52
53
54④ public Ranking() {
55     p0 = new JPanel();
56     p1 = new JPanel(new GridLayout(10, 2));
57     p2 = new JPanel();
58     p3 = new JPanel();
59     p4 = new JPanel();
60     exit = new JButton("Exit");
61     p2.add(exit);
62     panel1 = new JPanel() {
63         private BufferedImage image;
64         private int newWidth = 1250;
65         private int newHeight = 400;
66
67④         @Override
68         protected void paintComponent(Graphics g) {
69             super.paintComponent(g);
70             try {
71                 image = ImageIO.read(new File("./imagenes/portada.jpg"));
72                 g.drawImage(image.getScaledInstance(newWidth, newHeight, Image.SCALE_SMOOTH), 0, 0, null);
73             } catch (IOException e) {
74                 e.printStackTrace();
75             }
76         }
77     };
78
79④     exit.addActionListener(new ActionListener() {
80
81④         @Override
82         public void actionPerformed(ActionEvent e) {

```

```

80
81④         @Override
82         public void actionPerformed(ActionEvent e) {
83             //Botón para cerrar la pestaña de ranking.
84             dispose();
85
86         });
87
88         subtítulo = new JLabel("-----TOP 10 PLAYERS-----");
89         Font font1 = new Font("Arial", Font.BOLD, 30);
90         subtítulo.setFont(font1);
91
92         p4.add(subtítulo);
93         creators = new JLabel("By Cristian Alvarez | Alex Martinez | Patricio");
94         creators.setFont(new Font("Abyssinica SIL", Font.ITALIC, 10));
95
96         try {
97             //Creamos conexión a la base de datos para obtener los 10 players con más puntuación.
98             Class.forName("com.mysql.cj.jdbc.Driver");
99             Connection conn = DriverManager.getConnection(urlDatos, usuario, pass);
100            String query = "select * from PLAYERS order by Total_Points desc limit 10;";
101            PreparedStatement ps = conn.prepareStatement(query);
102            ResultSet rs = ps.executeQuery();
103
104            while(rs.next()) {
105                //Añadimos esos players y sus puntuaciones a labels.
106                JLabel label = new JLabel(rs.getString("player_name"));
107                JLabel label1 = new JLabel(Integer.toString(rs.getInt("Total_points")));
108                Font font3 = new Font("Arial", Font.PLAIN, 20);
109                label.setFont(font3);
110                label1.setFont(font3);
111                p1.add(label);
112                p1.add(label1);
113            }
114
115
116        } catch (ClassNotFoundException x) {
117            System.out.println("El driver no se ha cargado con éxito");
118        } catch (SQLException y) {
119            System.out.println(y.getMessage());
120        }
121        System.out.println("Conexión no creada con éxito!!!");

```

Console X
<terminated> MainWindow [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (17 may 2023, 20:41:12 – 20:41:24) [pid: 19376]

```

7         JLabel label = new JLabel(rs.getString("Player_name"));
8         JLabel label1 = new JLabel(Integer.toString(rs.getInt("Total_points")));
9         Font font3 = new Font("Arial", Font.PLAIN, 20);
0         label.setFont(font3);
1         label1.setFont(font3);
2         p1.add(label);
3         p1.add(label1);
4     }
5
6     } catch (ClassNotFoundException x) {
7         System.out.println("EL driver no se a cargado con exito");
8     } catch (SQLException y) {
9         System.out.println(y.getMessage());
0         System.out.println("Conexion no creada con exito!!!");
1     }
2
3
4     p0.setLayout(new BoxLayout(p0, BoxLayout.Y_AXIS));
5
6     p0.add(panel1);
7     p0.add(p4);
8     p0.add(p1);
9     p3.add(creators);
0     p0.add(p2);
1     p0.add(p3);
2     p0.setLayout(null);
3
4     panel1.setBounds(0, 0, 1250, 300);
5     p4.setBounds(0,300,1250,50);
6     p1.setBounds(520, 380, 300, 280);
7     p2.setBounds(0, 705, 1250, 30);
8     p3.setBounds(0, 740, 1258, 50);
9     this.add(p0);
this.setTitle("Ranking");
setDefaultCloseOperation(EXIT_ON_CLOSE);
setLocationRelativeTo(null);
setVisible(true);
7 }
3

```

Clase "Rounds":

Esta clase sirve para crear instancias de las rondas y añadirlas en un arraylist para así después facilitar la subida de datos a la base de datos.

```
orEnemy.java  Rounds.java X

public class Rounds {
    private int rounds_id;
    private int battle_id;
    private int warrior_id;
    private int warrior_weapon_id;
    private int opponent_id;
    private int opponent_weapon_id;
    private int injuries_Caused;
    private int injuries_Suffered;

    public Rounds(int rounds_id,int battle_id, int warrior_id, int warrior_weapon_id, int opponent_id, int opponent_weapon_id,
                 int injuries_Caused, int injuries_Suffered) {
        super();
        this.rounds_id = rounds_id;
        this.battle_id=battle_id;
        this.warrior_id = warrior_id;
        this.warrior_weapon_id = warrior_weapon_id;
        this.opponent_id = opponent_id;
        this.opponent_weapon_id = opponent_weapon_id;
        this.injuries_Caused = injuries_Caused;
        this.injuries_Suffered = injuries_Suffered;
    }

    public int getRounds_id() {
        return rounds_id;
    }

    public void setRounds_id(int rounds_id) {
        this.rounds_id = rounds_id;
    }

    public int getWarrior_id() {
        return warrior_id;
    }

    public void setWarrior_id(int warrior_id) {
        this.warrior_id = warrior_id;
    }

    public int getWarrior_weapon_id() {
        return warrior_weapon_id;
    }
```

```
import enemy.java | Rounds.java X
public int getWarrior_weapon_id() {
    return warrior_weapon_id;
}

public void setWarrior_weapon_id(int warrior_weapon_id) {
    this.warrior_weapon_id = warrior_weapon_id;
}

public int getOpponent_id() {
    return opponent_id;
}

public void setOpponent_id(int opponent_id) {
    this.opponent_id = opponent_id;
}

public int getOpponent_weapon_id() {
    return opponent_weapon_id;
}

public void setOpponent_weapon_id(int opponent_weapon_id) {
    this.opponent_weapon_id = opponent_weapon_id;
}

public int getInjuries_Caused() {
    return injuries_Caused;
}

public void setInjuries_Caused(int injuries_Caused) {
    this.injuries_Caused = injuries_Caused;
}

public int getInjuries_Suffered() {
    return injuries_Suffered;
}

public void setInjuries_Suffered(int injuries_Suffered) {
    this.injuries_Suffered = injuries_Suffered;
}
```

Clase "WarriorEnemy":

Es una clase que sirve para crear aleatoriamente a tu oponente.

```
1④ import java.sql.Connection;
9
10 public class WarriorEnemy {
11     private String urlDatos = "jdbc:mysql://localhost/RacesPAC?serverTimezone=UTC";
12     private String usuario = "root";
13     private String pass = "1234";
14     private ArrayList<Integer> arrayids;
15     private int num1,num2;
16     private Warrior warrior_enemy;
17     private Weapons weapon;
18④ WarriorEnemy(){
19 }
20 //Metodo para generar los enemigos de forma aleatoria
21④ public void Enemy_Random() {
22     try {
23         Random random = new Random();
24         arrayids = new ArrayList<Integer>();
25         WarriorContainer w1 = new WarriorContainer();
26         WeaponsContainer w2 = new WeaponsContainer();
27         w2.addWeapons();
28         w1.addWarrior();
29         //
30         num2 = random.nextInt(w1.getWarriorarray().size());
31         warrior_enemy = w1.getWarriorarray().get(num2);
32         Class.forName("com.mysql.cj.jdbc.Driver");
33         Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
34         String query = "select Weapon_id from WEAPONS_AVAILABLE where Warrior_id =" +warrior_enemy.getId();
35         PreparedStatement ps = conn.prepareStatement(query);
36         ResultSet rs = ps.executeQuery();
37         while(rs.next()) {
38             arrayids.add(rs.getInt(1));
39         }
40         num1 = (int)(Math.random()*(arrayids.size()));
41         for (Weapons weapons1:w2.getWeaponsarray()) {
42             if(arrayids.get(num1)==weapons1.getId()) {
43                 weapon = weapons1;
44                 break;
45             }
46         }
47
48
49 }
```

```

5         PreparedStatement ps = conn.prepareStatement(query);
6         ResultSet rs = ps.executeQuery();
7         while(rs.next()) {
8             arrayids.add(rs.getInt(1));
9         }
10        num1 = (int)(Math.random()*(arrayids.size()));
11        for (Weapons weapons1:w2.getWeaponsarray()) {
12            if(arrayids.get(num1)==weapons1.getId()) {
13                weapon = weapons1;
14                break;
15            }
16        }
17    }
18
19 } catch (ClassNotFoundException x) {
20     System.out.println("EL driver no se a cargado con exito");
21 } catch (SQLException y) {
22     System.out.println(y.getMessage());
23     System.out.println("Conexion no creada con exito!!!");
24 }
25
26 public Warrior getWarrior_enemy() {
27     return warrior_enemy;
28 }
29
30 public void setWarrior_enemy(Warrior warrior_enemy) {
31     this.warrior_enemy = warrior_enemy;
32 }
33
34 public Weapons getWeapon() {
35     return weapon;
36 }
37
38 public void setWeapon(Weapons weapon) {
39     this.weapon = weapon;
40 }
41 }
42

```

Clase "BBDD":

Esta clase contiene varios métodos que se utilizan para subir los datos del usuario, las rondas y las batallas de cada partida.

```

1# import java.awt.Image;
2
3 public class BBDD {
4     private String urlDatos = "jdbc:mysql://localhost/RacesPAC?serverTimezone=UTC";
5     private String usuario = "root";
6     private String pass = "1234";
7
8     BBDD(){
9
10 }
11
12 //Metodo para añadir las rondas de una batalla a la base de datos.
13 public void addRounds(ArrayList<Rounds> array) {
14     try {
15         Class.forName("com.mysql.cj.jdbc.Driver");
16         Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
17         String query="INSERT INTO ROUNDS (Round_Number,Battle_id,Warrior_Weapon_id,Opponent_id,Opponent_Weapon_id,Injuries_Caused,Injuries_Suffered) values (?, ?, ?, ?, ?, ?, ?)";
18         PreparedStatement ps = conn.prepareStatement(query);
19         for (Rounds insert:array) {
20             ps.setInt(1, insert.getRounds_id());
21             ps.setInt(2, battleId());
22             ps.setInt(3, insert.getWarrior_id());
23             ps.setInt(4, insert.getWarrior_weapon_id());
24             ps.setInt(5, insert.getOpponent_id());
25             ps.setInt(6, insert.getOpponent_weapon_id());
26             ps.setInt(7, insert.getInjuries_caused());
27             ps.setInt(8, insert.getInjuries_suffered());
28             ps.executeUpdate();
29         }
30     } catch (ClassNotFoundException x) {
31     System.out.println("EL driver no se a cargado con exito");
32 } catch (SQLException y) {
33     System.out.println(y.getMessage());
34     System.out.println("Conexion no creada con exito!!!");
35 }
36 }
37
38 //Metodo para insertar una nueva batalla.
39 public void insertBattle(int player_id,int warrior_id,int Warrior_Weapon_id,int Opponent_id,int Opponent_Weapon_id, int Injuries_Caused, int Injuries_Suffered, int Battle_Points) {
40     try {
41         Connection conn = DriverManager.getConnection(urlDatos, usuario, pass);
42         String ouerv = "insert into BATTLE (Player_id,Warrior_id,Warrior_Weapon_id,Oponent_id,Oponent_Weapon_id,Injuries_Caused,Injuries_Suffered,Battle_Points) values (?, ?, ?, ?, ?, ?, ?, ?)";
43     }
44 }
45
46
47
48
49
50
51

```

```

49     try {
50         Connection conn = DriverManager.getConnection(urlDatos, usuario, pass);
51         String query = "insert into BATTLE (Player_id,Warrior_id,Warrior_Weapon_id,Opponent_id,Opponent_Weapon_id,Injuries_Caused,Injuries_Suffered,Battle_Points) values (?,?,?,?,?,?,?,?)";
52         PreparedStatement ps = conn.prepareStatement(query);
53         ps.setInt(1, player_id);
54         ps.setInt(2, warrior_id);
55         ps.setInt(3, Warrior_Weapon_id);
56         ps.setInt(4, Opponent_id);
57         ps.setInt(5, Opponent_Weapon_id);
58         ps.setInt(6, Injuries_Caused);
59         ps.setInt(7, Injuries_Suffered);
60         ps.setInt(8, Battle_Points);
61         ps.executeUpdate();
62     } catch (SQLException y) {
63         System.out.println(y.getMessage());
64         System.out.println("Conexion no creada con exito!!!");
65     }
66 }
67
68 //Metodo para coger el id de la batalla, para poder pasarselo al metodo de añadir rondas.
69 public int battleId() {
70     int id=0;
71     try {
72         Connection conn = DriverManager.getConnection(urlDatos, usuario, pass);
73         Statement stmt = conn.createStatement();
74         String query = "SELECT max(battle_id) from BATTLE";
75         ResultSet rs = stmt.executeQuery(query);
76
77         if (rs.next()) {
78             id = rs.getInt(1);
79         }
80     } catch (SQLException y) {
81         System.out.println(y.getMessage());
82         System.out.println("Conexion no creada con exito!!!");
83     }
84     return id;
85 }
86
87
88 //Metodo para coger el player_id.
89 public int playerId() {
90
91     //Metodo para coger el player_id.
92     public int playerId() {
93         int id=-1;
94         try {
95             Connection conn = DriverManager.getConnection(urlDatos, usuario, pass);
96             Statement stmt = conn.createStatement();
97             String query = "SELECT max(player_id) from PLAYERS";
98             ResultSet rs = stmt.executeQuery(query);
99
100            if (rs.next()) {
101                id = rs.getInt(1);
102            }
103        } catch (SQLException y) {
104            System.out.println(y.getMessage());
105            System.out.println("Conexion no creada con exito!!!");
106        }
107        return id;
108    }
109
110
111 //Metodo para insertar un nuevo player.
112 public void insertPlayer(String username) {
113     try {
114         Class.forName("com.mysql.cj.jdbc.Driver");
115         Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
116         String query = "insert into PLAYERS (Player_name,Total_Points) values (?,?)";
117         PreparedStatement ps = conn.prepareStatement(query);
118         ps.setString(1, username);
119         ps.setInt(2, 0);
120         ps.executeUpdate();
121     } catch (ClassNotFoundException x) {
122         System.out.println("EL driver no se a cargado con exito");
123     } catch (SQLException y) {
124         System.out.println(y.getMessage());
125         System.out.println("Conexion no creada con exito!!!");
126     }
127 }
128
129 //Metodo para hacer un update del player. sirve para ponerle los puntos.

```

```

109 //Metodo para insertar un nuevo player.
110 public void insertPlayer(String username) {
111     try {
112         Class.forName("com.mysql.cj.jdbc.Driver");
113         Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
114         String query = "insert into PLAYERS (Player_name,Total_Points) values (?,?);";
115         PreparedStatement ps = conn.prepareStatement(query);
116         ps.setString(1, username);
117         ps.setInt(2, 0);
118         ps.executeUpdate();
119     } catch (ClassNotFoundException x) {
120         System.out.println("EL driver no se a cargado con exito");
121     } catch (SQLException y) {
122         System.out.println(y.getMessage());
123         System.out.println("Conexion no creada con exito!!!");
124     }
125 }
126
127
128 //Metodo para hacer un update del player, sirve para ponerle los puntos.
129 public void updatePlayer(String username,int total_points) {
130     try {
131         Class.forName("com.mysql.cj.jdbc.Driver");
132         Connection conn = DriverManager.getConnection(urlDatos,usuario,pass);
133         String query = "update PLAYERS set Total_Points = ? where Player_name = ?";
134         PreparedStatement ps = conn.prepareStatement(query);
135         ps.setInt(1,total_points);
136         ps.setString(2, username);
137         ps.executeUpdate();
138     } catch (ClassNotFoundException x) {
139         System.out.println("EL driver no se a cargado con exito");
140     } catch (SQLException y) {
141         System.out.println("Conexion no creada con exito!!!");
142     }
143 }
144
145
146 }
147

```

El uso de Java como lenguaje de programación principal nos ha permitido desarrollar una mecánica de juego sólida y flexible, brindando una experiencia interactiva y entretenida para los jugadores de "*THE BATTLE FOR MIDDLE-EARTH*".

Glosario de términos

- **Clase**

En programación, una clase es un modelo o plantilla que define las propiedades y comportamientos de un objeto. En Java, las clases se utilizan para crear objetos y encapsular la lógica del programa.

- **Métodos**

En el contexto de la programación orientada a objetos, los métodos son bloques de código que se definen en una clase y se utilizan para realizar operaciones específicas. Los métodos pueden tener parámetros y devolver resultados.

- **Base de datos**

Una base de datos es un sistema de almacenamiento organizado de información. Se utiliza para almacenar y recuperar datos de manera eficiente. En el contexto del proyecto, se utilizó una base de datos para almacenar información relacionada con los usuarios, partidas y atributos del juego.

- **Interfaz gráfica**

Una interfaz gráfica de usuario (GUI, por sus siglas en inglés) es una forma de interactuar con una aplicación utilizando elementos visuales, como botones, ventanas y menús. Proporciona una forma intuitiva y visual para que los usuarios interactúen con el software.

- **GitHub**

GitHub es una plataforma en línea que facilita el alojamiento y la colaboración en proyectos de desarrollo de software. Permite a los desarrolladores trabajar juntos, gestionar versiones de código y realizar un seguimiento de los cambios en un repositorio.

- **Razas**

En el contexto del juego "Batallas por la Tierra Media", las razas se refieren a los grupos o especies de personajes disponibles para jugar. En este caso, las razas mencionadas son humanos, enanos y elfos.

Errores Comunes

Cuando juegues nuestro juego, es posible que te encuentres con algunos errores comunes. Por ejemplo, si eliges un nombre de usuario extremadamente largo, es probable que ocurra un error porque hay demasiados caracteres. Para evitar este problema, te recomendamos usar un nombre de usuario de longitud moderada (alrededor de 30 caracteres).

Otro error que podrías encontrar es cuando ves la lista de los mejores puntajes. Si algunos nombres de los jugadores son muy largos, es posible que algunas letras se corten y se muestren tres puntos suspensivos en su lugar. Esto sucede para indicar que no se muestra el nombre completo. Una solución simple sería usar nombres más cortos o abreviaciones.

Además, es importante asegurarse de que los ajustes de la base de datos estén correctamente configurados para que funcione correctamente. Si los ajustes no son correctos, podrías tener problemas al guardar o recuperar información del juego. Te sugerimos verificar cuidadosamente la configuración de la base de datos y corregir cualquier error que encuentres. Las credenciales de acceso de la base de datos local y las escritas en el código deben coincidir.

También, debes asegurarte de que el módulo encargado de obtener información de la base de datos esté implementado correctamente. Si hay errores en su implementación, podría afectar el funcionamiento del juego, especialmente al intentar obtener datos importantes. Asegurarse que se están utilizando las herramientas de **mysql-connector-java-8.0.15.jar** en el proyecto del programa java.

FAQ

P: ¿Cuáles son los mayores desafíos al implementar la mecánica de un juego en Java?

R: La implementación de la mecánica de un juego en Java puede ser un desafío debido a la gran cantidad de variables que hay que tener en cuenta. A medida que el juego se vuelve más complejo, es necesario gestionar correctamente las interacciones entre personajes, las reglas de combate, los elementos del juego, entre otros aspectos. La planificación cuidadosa y el diseño modular del código pueden ayudar a abordar estos desafíos.

P: ¿Qué problemas comunes se pueden encontrar al trabajar con una base de datos en Java?

R: Al trabajar con una base de datos en Java, es posible encontrarse con problemas de conectividad y acceso a la base de datos. En ocasiones, la conexión puede fallar debido a problemas de configuración, credenciales incorrectas o problemas de red. También puede haber errores al escribir consultas SQL o al pasar los datos de la base de datos a la aplicación. La comprensión de los conceptos básicos de las bases de datos y la depuración efectiva son habilidades importantes para resolver estos problemas.

P: ¿Cómo puedo solucionar problemas de conectividad con una base de datos en Java?

R: Si estás experimentando problemas de conectividad con una base de datos en Java, hay algunas acciones que puedes tomar. Primero, verifica la configuración de conexión, asegurándote de que los detalles de la base de datos (nombre de usuario, contraseña, URL) sean correctos. Asegúrate de que el servidor de la base de datos esté en funcionamiento y accesible desde tu aplicación. Si el problema persiste, revisa los registros

de errores y los mensajes de excepción para obtener pistas sobre la causa del problema y realiza las correcciones necesarias.

P: ¿Qué estrategias se pueden utilizar para gestionar la complejidad en un proyecto de programación en Java?

R: Al lidiar con la complejidad en un proyecto de programación en Java, es útil utilizar técnicas como la modularización y la abstracción. Dividir el código en clases y métodos cohesivos ayuda a organizar la lógica y facilita su comprensión y mantenimiento. Además, es importante que durante la escritura del código se dejen comentarios a modo de documentación para hacer la revisión a futuro mucho más sencilla. Además de permitir a nuevos integrantes del equipo entender el funcionamiento detrás de la lógica del programa.

Webgrafía

- **Stack Overflow (<https://stackoverflow.com>)**

Una comunidad en línea donde puedes hacer preguntas y encontrar respuestas sobre programación. Es una excelente fuente de conocimiento y resolución de problemas.

- **GitHub (<https://github.com>)**

Una plataforma de alojamiento de repositorios de código fuente. Aquí encontrarás proyectos de código abierto, bibliotecas y herramientas que te pueden ser útiles en tus proyectos.

- **W3Schools (<https://www.w3schools.com>)**

Un sitio web con tutoriales, documentación y ejemplos de código para aprender y consultar diferentes tecnologías web, como HTML y CSS en nuestro caso.

- **Mozilla Developer Network (<https://developer.mozilla.org>)**

Una completa documentación y guía de referencia para tecnologías web, incluyendo HTML y CSS.

- **Java Documentation (<https://docs.oracle.com/javase/>)**

La documentación oficial de Java proporcionada por Oracle. Aquí encontrarás la documentación completa de las clases y métodos de Java, así como guías de programación.

- **JetBrains (<https://www.jetbrains.com>)**

Un conjunto de herramientas y entornos de desarrollo integrados (IDE) para diferentes lenguajes de programación, como IntelliJ IDEA para Java, PyCharm para Python y muchos más.

- **Chat GPT (OpenAI)**

El Chat GPT de OpenAI puede ser una fuente útil de información y ayuda en la programación o a la hora de entender conceptos.