

# GUÍA USUARIO - JUEGO MAZMORRAS



*Alejandro Rey Tostado*

*Alberto García Izquierdo*

*Programación*

*DAW 1ºA*

## **ÍNDICE**

<b>1. Carpeta Modelo</b>	<b>3</b>
<b>1.1. Personaje</b>	<b>3</b>
<b>1.2. Protagonista</b>	<b>5</b>
<b>1.3. Enemigo</b>	<b>5</b>
<b>1.4. Celda</b>	<b>6</b>
<b>1.5. Mapa</b>	<b>7</b>
<b>2. Carpeta Controladores</b>	<b>8</b>
<b>2.1. Controlador Personaje</b>	<b>8</b>
<b>2.2. Controlador Juego</b>	<b>8</b>
<b>3. Carpeta Recursos</b>	<b>9</b>
<b>4. Carpeta Vistas</b>	<b>10</b>
<b>4.1. Creacion Personaje</b>	<b>10</b>
<b>4.2. Juego</b>	<b>10</b>
<b>5. Clase Principal</b>	<b>11</b>
<b>6. Funcionamiento del Juego</b>	<b>12</b>

## 1. Carpeta Modelo

Contiene las clases principales que forman la lógica del juego:

### 1.1. Personaje

Clase abstracta que define los atributos comunes a todos los personajes del juego: nombre, posición, salud, fuerza, defensa, velocidad y percepción. También contiene métodos como recibirDanio, atacar, y getters/setters.

```
juego_mazmorras > src > main > java > com > alejandro > alberto > modelo > Personaje.java > Personaje >
ElReeeyy, 6 hours ago | 2 authors (ElReeeyy and one other)
1 package com.alejandro.alberto.modelo;
2
3 ElReeeyy, 6 hours ago | 2 authors (ElReeeyy and one other)
4 /**
5  * Clase abstracta que define un personaje del juego
6  *
7  * @author Alejandro Rey Tostado y Alberto García Izquierdo
8  */
9 public abstract class Personaje {
10     protected String nombre;
11     protected int salud;
12     protected int fuerza;
13     protected int defensa;
14
15     protected int x;
16     protected int y;
17
18     /**
19      * Constructor para todos los personajes
20      *
21      * @param nombre nombre del personaje
22      * @param salud salud del personaje
23      * @param fuerza fuerza del personaje
24      * @param defensa defensa del personaje
25      */
26     public Personaje(String nombre, int salud, int fuerza, int defensa) {
27         this.nombre = nombre;
28         this.salud = salud;
29         this.fuerza = fuerza;
30         this.defensa = defensa;
31     }
32 }
```

```

// METODOS
/**
 * Mueve al personaje a una nueva posicion
 *
 * @param x nueva posición en el eje X
 * @param y nueva posición en el eje Y
 */
public void setPosicion(int x, int y) {
    this.x = x;
    this.y = y;
}

/**
 * Metodo en el que se introduce la cantidad de daño que ha recibido el personaje y se le resta
 *
 * @param cantidad cantidad de daño recibido
 */
public void recibirDanio(int cantidad) {
    if (defensa > 0) {
        int restante = cantidad - defensa;
        defensa -= cantidad;
        if (defensa < 0) defensa = 0;

        if (restante > 0) {
            salud -= restante;
        }
    } else {
        salud -= cantidad;
    }

    if (salud < 0) salud = 0;

    System.out.println(nombre + " - Salud: " + salud + ", Defensa: " + defensa);

    if (!estaVivo()) {
        System.out.println(nombre + " ha muerto");
    }
}

```

```

/**
 * Comprueba si el personaje esta vivo o no
 *
 * @return TRUE si el personaje esta vivo, FALSE si no lo esta
 */
public boolean estaVivo() {
    return this.salud > 0;
}

// GETTERS Y SETTERS

public String getNombre() {
    return this.nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public int getSalud() {
    return this.salud;
}

public void setSalud(int salud) {
    this.salud = salud;
}

public int getFuerza() {
    return this.fuerza;
}

public void setFuerza(int fuerza) {
    this.fuerza = fuerza;
}

public int getDefensa() {
    return this.defensa;
}

public void setDefensa(int defensa) {
    this.defensa = defensa;
}

```

## 1.2. Protagonista

Extiende de Personaje. Representa al jugador principal. Se crea desde la pantalla de creación introduciendo nombre, salud y fuerza. Se controla con las flechas del teclado durante la partida.

```
package com.alejandro.alberto.modelo;

You, 5 days ago | 2 authors (ElReeey and one other)
/**
 * Clase que representa al protagonista en el juego.
 * Hereda de la clase Personaje.
 *
 * @author Alejandro Rey Tostado y Alberto García Izquierdo
 */
public class Protagonista extends Personaje {

    /**
     * Constructor para el protagonista
     *
     * @param nombre nombre del protagonista
     * @param salud salud del protagonista
     * @param fuerza fuerza del protagonista
     * @param defensa defensa del protagonista
     */
    public Protagonista(String nombre, int salud, int fuerza, int defensa) {
        super(nombre, salud, fuerza, defensa);
    }
}

ElReeey, 3 weeks ago • Creacion personajes y derivados
```

## 1.3. Enemigo

También extiende de Personaje. Representa a los enemigos generados en el mapa. Tienen comportamiento automático: se mueven, atacan y persiguen al jugador si está cerca.

```
package com.alejandro.alberto.modelo;

You, 8 hours ago | 2 authors (ElReeey and one other)
/**
 * Clase que representa a un enemigo en el juego.
 * Hereda de la clase Personaje.
 *
 * @author Alejandro Rey Tostado y Alberto García Izquierdo
 */
public class Enemigo extends Personaje {
    private int percepcion; // rango de vision del enemigo

    /**
     * Constructor para los enemigos
     *
     * @param salud salud del enemigo
     * @param fuerza fuerza del enemigo
     * @param defensa defensa del enemigo
     * @param percepcion rango de vision del enemigo
     */
    public Enemigo(int salud, int fuerza, int defensa, int percepcion) {
        super(nombre:"Enemigo", salud, fuerza, defensa); // le asignamos un nombre por defecto
        this.percepcion = percepcion;
    }

    // GETTERS Y SETTERS
    public int Percepcion() {
        return this.percepcion;
    }

    public void setPercepcion(int percepcion) {
        this.percepcion = percepcion;
    }

    private int velocidad;

    public int getVelocidad() {
        return velocidad;
    }

    public int getPercepcion() {
        return percepcion;
    }
}

ElReeey, 3 weeks ago • Creacion personajes y derivados ...
```

#### 1.4. Celda

Representa una casilla del mapa. Cada celda puede ser de tipo "suelo" o "pared", y puede contener a un personaje (protagonista o enemigo). Se usa para modelar el escenario lógicamente.

```
package com.alejandro.alberto.modelo;

You, 3 days ago | 2 authors (ElReeey and one other)
/**
 * Esta clase representa una celda individual en el mapa del juego.
 * Cada celda puede ser un suelo o una pared.
 *
 * @author Alberto García Izquierdo y Alejandro Rey Tostado
 */
public class Celda {
    private String tipo; // puede ser "suelo" o "pared"
    private int fila;
    private int columna;
    private Personaje personaje; // Personaje que ocupa la celda, si lo hay

    /**
     * Crea una celda con la posición y el tipo
     *
     * @param tipo "suelo" o "pared"
     * @param fila número de fila de la celda
     * @param columna número de columna de la celda
     */
    public Celda(String tipo, int fila, int columna) {
        this.tipo = tipo;
        this.fila = fila;
        this.columna = columna;
        this.personaje = null; // Inicialmente no hay personaje
    }
}
```

```
// Metodos
/**
 * Verifica si la celda está ocupada por un personaje.
 *
 * @return true si hay un personaje, false en caso contrario
 */
public boolean estaOcupada() {
    return this.personaje != null;
}

public boolean esTransitable() {
    return tipo.equals(anObject:"suelo") && !estaOcupada();
}
You, 3 days ago * actu

public boolean contieneEnemigo() {
    return estaOcupada() && !personaje.esProtagonista();
}

// Getters y Setters

public String getTipo() {
    return this.tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public int getFila() {
    return this.fila;
}

public void setFila(int fila) {
    this.fila = fila;
}

public int getColumna() {
    return this.columna;
}

public void setColumna(int columna) {
    this.columna = columna;
}
}
```

## 1.5. Mapa

Clase que representa el escenario del juego. Se encarga de cargar el mapa desde un archivo .txt (escenario.txt) que contiene letras P para pared y S para suelo. Convierte esta información en una matriz de objetos Celda[][].

```
package com.alejandro.alberto.modelo;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

You, 3 days ago | 2 authors (You and one other)
/**
 * Clase que representa el escenario del juego.
 * Usa una matriz de caracteres donde 'P' es pared y 'S' es suelo.
 *
 * @author Alejandro Rey Tostado y Alberto García Izquierdo.
 */
public class Mapa {

    private Celda[][] celda;

    /**
     * Constructor: carga el mapa desde un archivo de texto.
     *
     * @param rutaArchivo Ruta del archivo del mapa.
     */
    public Mapa(String rutaArchivo) throws IOException {
        cargarMapa(rutaArchivo);
    }
}
```

```
/**
 * Carga el mapa desde un archivo de texto.
 * El archivo debe contener 'P' para paredes y 'S' para suelos.
 *
 * @param rutaArchivo Ruta del archivo del mapa.
 */
private void cargarMapa(String rutaArchivo) {
    try (BufferedReader br = new BufferedReader(new FileReader(rutaArchivo))) {
        // Leer todas las líneas
        List<String> lineas = new ArrayList<>();
        String linea;
        while ((linea = br.readLine()) != null) {
            lineas.add(linea);
        }

        int filas = lineas.size();
        int columnas = lineas.get(index:0).split(regex:" ").length;

        celda = new Celda[filas][columnas];

        for (int i = 0; i < filas; i++) {
            String[] elementos = lineas.get(i).split(regex:" ");
            for (int j = 0; j < columnas; j++) {
                String tipo = elementos[j].equals(anObject:"P") ? "pared" : "suelo";
                celda[i][j] = new Celda(tipo, i, j);
            }
        }
    } catch (IOException e) {
        System.err.println("Error al cargar el mapa: " + e.getMessage());
    }
}

public Celda[][] getCeldas() {
    return this.celda;
}

public void setCelda(Celda[][] celda) {
    this.celda = celda;
}
}
```

## 2. Carpeta Controladores

Contiene las clases que gestionan la lógica entre el modelo y la interfaz del usuario (FXML).

### 2.1. Controlador Personaje

Controla la pantalla de creación del personaje. Lee el nombre, salud y fuerza introducidos por el usuario. Cuando se pulsa el botón "Comenzar Juego", se cambia a la pantalla del juego (Juego.fxml) y se inicializan los datos del protagonista.

```
@FXML
public void comenzar(ActionEvent event) {
    try {
        String nombrePersonaje = nombre.getText();
        if (nombrePersonaje.isEmpty()) {
            mostrarAlerta(titulo:"Error", mensaje:"Por favor, ingresa un nombre para tu personaje.");
            return;
        }

        int saludPersonaje = salud.getText().isEmpty() ? 100 : Integer.parseInt(salud.getText());
        int fuerzaPersonaje = fuerza.getText().isEmpty() ? 10 : Integer.parseInt(fuerza.getText());

        if (saludPersonaje <= 0 || fuerzaPersonaje <= 0) {
            mostrarAlerta(titulo:"Error", mensaje:"Salud y fuerza deben ser valores positivos.");
            return;
        }

        Protagonista protagonista = new Protagonista(nombrePersonaje, saludPersonaje, fuerzaPersonaje, defensa:5);
        App.mostrarVistaJuego(protagonista);
    } catch (NumberFormatException e) {
        mostrarAlerta(titulo:"Error", mensaje:"Por favor ingresa valores numéricos válidos.");
    } catch (Exception e) {
        mostrarAlerta(titulo:"Error", mensaje:"Error al iniciar el juego: " + e.getMessage());
    }
}

private void mostrarAlerta(String titulo, String mensaje) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle(titulo);
    alert.setHeaderText(null);
    alert.setContentText(mensaje);
    alert.showAndWait();
}
```

### 2.2. Controlador Juego

Es la base del juego. Controla el movimiento del jugador con el teclado, el combate, el turno de los enemigos, la detección de fin de partida, y el dibujo del tablero en el GridPane. También actualiza las estadísticas del jugador en pantalla.



```

public class controladorJuego {
    /**
     * Método para manejar las teclas presionadas
     *
     * @param event Evento de la tecla presionada
     */
    @FXML
    public void manejarTeclas(KeyEvent event) {
        if (!protagonista.estaVivo() || juegoTerminado) return;

        switch (event.getCode()) {
            case UP: moverProtagonista(dx:0, -1); break;
            case DOWN: moverProtagonista(dx:0, dy:1); break;
            case LEFT: moverProtagonista(dx:-1, dy:0); break;
            case RIGHT: moverProtagonista(dx:1, dy:0); break;
            default: return;
        }

        moverEnemigos();
        actualizarEstadisticas();
        dibujarTablero();

        if (!protagonista.estaVivo()) {
            juegoTerminado = true;
            mostrarGameOver();
        } else if (enemigos.stream().noneMatch(enemigo:estaVivo)) {
            juegoTerminado = true;
            mostrarVictoria();
        }
    }

    /**
     * Método para mover al protagonista
     *
     * @param dx Cambio en la posición X
     * @param dy Cambio en la posición Y
     */
    private void moverProtagonista(int dx, int dy) {
        int nuevaX = protagonista.getX() + dx;
        int nuevaY = protagonista.getY() + dy;

        if (nuevaX < 0 || nuevaY < 0 || nuevaY >= mapa.size() || nuevaX >= mapa.get(nuevaY).length()) return;
        if (mapa.get(nuevaY).charAt(nuevaX) == '#') return;

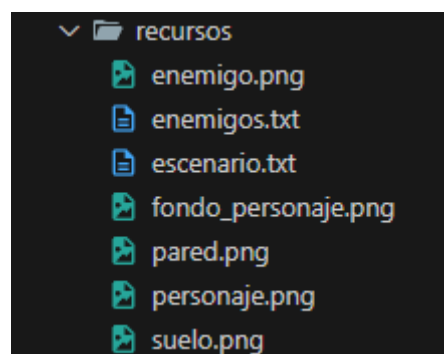
        for (Enemigo enemigo : enemigos) {
            if (enemigo.getX() == nuevaX && enemigo.getY() == nuevaY && enemigo.estaVivo()) {
                protagonista.atacar(enemigo);
                actualizarEstadisticas();
            }
        }
    }
}

```

### 3. Carpeta Recursos

Contiene todos los recursos necesarios para que el juego funcione correctamente:

- *Escenario.txt*: archivo de texto que representa el tablero del juego mediante letras ("#" = pared, "-" = suelo).
- *Enemigos.txt*: archivo de texto donde se carga las posiciones de los enemigos.
- *Protagonista.png*: imagen que representa al jugador.
- *Enemigo.png*: imagen que representa a los enemigos.
- *Fondo\_Personaje.png*: imagen que representa el fondo del juego.
- *Pared.png*: imagen que representa el muro del tablero de juego.
- *Suelo.png*: imagen que representa el suelo del tablero de juego.



## 4. Carpeta Vistas

Aquí están los archivos “.fxml” diseñados con SceneBuilder para construir la interfaz gráfica.

### 4.1. Creacion Personaje

Pantalla inicial donde el usuario introduce el nombre, salud y fuerza del protagonista. Tiene un diseño simple y está ambientado visualmente con un fondo de mazmorra. Al pulsar el botón, se pasa al juego.

```
<StackPane fx:id="root" prefHeight="512" prefWidth="768" xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="...">
    <!-- Imagen de fondo RPG -->
    <ImageView fx:id="fondo" fitHeight="512" fitWidth="768" preserveRatio="false">
        <image>
            <Image url="@../recursos/fondo_personaje.png" />
        </image>
    </ImageView>

    <!-- Contenido principal en horizontal -->
    <HBox alignment="CENTER" spacing="40">

        <!-- Imagen del personaje a la izquierda -->
        <VBox alignment="CENTER">
            <ImageView fx:id="personaje" fitHeight="195.0" fitWidth="139.0">
                <image>
                    <Image url="@../recursos/personaje.png" />
                </image>
            </ImageView>
        </VBox>

        <!-- Panel de entrada de datos -->
        <VBox alignment="CENTER_LEFT" prefHeight="512.0" prefWidth="488.0" spacing="15" style="-fx-padding: 20px; -fx-background-radius: 10;">
            <Label fx:id="textoPrin" text="CREACIÓN DE PERSONAJE" textFill="WHITE">
                <font>
                    <Font name="Bookman Old Style" size="22.0" />
                </font>
            </Label>

            <HBox alignment="CENTER_LEFT" spacing="10">
                <Label minWidth="80" style="-fx-font-weight: bold;" text="Nombre:" textFill="white">
                    <font>
                        <Font size="16.0" />
                    </font>
                </Label>
                <Label>
                    ElReesyyy, last week + cambios
                </Label>
                <TextField fx:id="nombre" prefWidth="180" promptText="Introduce tu nombre" />
            </HBox>
        </VBox>
    </HBox>
</StackPane>
```

### 4.2. Juego

Es la pantalla principal del juego. Contiene:

- Un GridPane donde se dibuja el tablero (mapa + personajes).
- Un panel lateral (VBox) que muestra las estadísticas del jugador: nombre, salud, fuerza, defensa.
- El evento `onKeyPressed` está asignado para permitir mover al jugador con las teclas.

```

<GridPane xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.alejandro.alberto.controladores.controladorJuego"
onKeyPressed="#manejarTeclas">
  <columnConstraints>
    <ColumnConstraints prefWidth="450.0" />
    <ColumnConstraints prefWidth="250.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints prefHeight="450.0" />
  </rowConstraints>

  <!-- Panel izquierdo: tablero 15x15 -->
  <GridPane fx:id="mapaGrid" hgap="0" vgap="0" GridPane.columnIndex="0" GridPane.rowIndex="0">
    <columnConstraints>
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
      <ColumnConstraints prefWidth="30.0" />
    </columnConstraints>
    <rowConstraints>
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
      <RowConstraints prefHeight="30.0" />
    </rowConstraints>
  </GridPane>

```

## 5. Clase Principal

### App.java

Es la clase que lanza la aplicación JavaFX. Carga el primer archivo FXML (CreacionPersonaje.fxml) y controla el cambio entre pantallas. Usa la clase FXMLLoader para enlazar vistas y controladores.

```

/**
 * Método principal de la aplicación
 *
 * @param args Argumentos de la línea de comandos
 */
@Override
public void start(Stage stage) throws IOException {
    primaryStage = stage;
    mostrarVistaCreacionPersonaje();
    stage.setTitle("Juego de Mazmorras - Creación de Personaje");
    stage.setResizable(false);
    stage.show();
}

/**
 * Método para mostrar la vista de creación de personaje
 *
 * @throws IOException Si ocurre un error al cargar la vista
 */
public static void mostrarVistaCreacionPersonaje() throws IOException {
    Parent root = FXMLLoader.load(App.class.getResource(name: "/com/alejandro/alberto/vistas/CreacionPersonaje.fxml"));
    primaryStage.setScene(new Scene(root, 768, 512));
}

/**
 * Método para mostrar la vista del juego
 *
 * @param protagonista El protagonista que se va a jugar
 * @throws IOException Si ocurre un error al cargar la vista
 */
public static void mostrarVistaJuego(Protagonista protagonista) throws IOException {
    protagonistaActual = protagonista;
    FXMLLoader loader = new FXMLLoader(App.class.getResource(name: "/com/alejandro/alberto/vistas/Juego.fxml"));
    Parent root = loader.load();

    controladorJuego controller = loader.getController();
    controller.setProtagonista(protagonistaActual);

    primaryStage.setScene(new Scene(root, 800, 600));
    root.requestFocus();
    primaryStage.setTitle("Juego de Mazmorras - " + protagonistaActual.getNombre());
}

Run | Debug
public static void main(String[] args) {
    launch(args);
}

```

## 6. Funcionamiento del Juego

### ● Inicio

1. El usuario abre la aplicación.
2. Se muestra la pantalla de creación de personaje.
3. El usuario introduce su nombre y selecciona valores iniciales de salud y fuerza.
4. Al pulsar "Comenzar Juego", se inicializa el mapa y se posicionan el protagonista y los enemigos.

### ● Jugabilidad

- El jugador se mueve con las flechas del teclado.
- El protagonista no puede atravesar paredes.
- Si intenta moverse a una casilla con un enemigo, lo ataca en su lugar.
- Después del turno del jugador, los enemigos se mueven automáticamente:

- Si ven al jugador (dentro de su percepción), lo persiguen.
- Si no, se mueven aleatoriamente.

- **Combate**

- Cada personaje tiene fuerza y salud.
- Si la salud de un personaje baja a 0, muere.
- El jugador puede atacar enemigos, y viceversa.

- **Fin del juego**

- Si el jugador muere → aparece un mensaje de "Has perdido".
- Si todos los enemigos son derrotados → aparece "Has ganado".