



Universidad de San Carlo de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la programación y computación 1
Ing. Moisés Velásquez
Auxiliar. Pablo Oliva

PRACTICA No.2: INFORME DE DESARROLLO

Guillermo Enrique Marroquin Morán

202103527

Guatemala 26 de ago. de 25

INDICE

Informe de desarrollo de la práctica 1	3
¿Cómo desarrolle la práctica?	3
¿Qué problemas tuve con la práctica?.....	17
¿Cómo se pudo solucionar los problemas.....	18
Menú interactivo	19
EGRÁFIA	27

Informe de desarrollo de la práctica 1

¿Cómo desarrolle la práctica?

Esta práctica se llevó a cabo con el lenguaje de java con el IDE de NetBeans, parte de esta tuvo un registro en git, subiéndose *commits* al repositorio en GitHub. Iniciando con el pie derecho con esta práctica, empecé leyendo muchas veces el enunciado, para tener en cuenta todos los puntos clave, los anoté en mi aplicación de confianza de notas, luego... empecé haciendo algo muy peculiar, la clase de pantalla de bienvenida, que bueno es eso, una bienvenida, con una canción de fondo, por lo que la “clase principal” se pasó a ser la clase del menú principal.

```
/*
public class Bienvenida extends JFrame {

    private MusicPlayer mPlayer; //quiero reproducir musica en la ventana de bienvenida

    public Bienvenida(String audiofilepath) {
        setTitle("Bienvenido a AREA USAC 2025"); //esto saldra en la ventana osea... arriba
        //tamano de la ventana
        setSize(700, 400);
        setLocationRelativeTo(null); // para centrar la ventana
        setUndecorated(true);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        //hagamos un boton pa salir... que dar click dobel y salir es muy aburrido
        JPanel mainPanel = new JPanel(new GridLayout(2, 1));

        //El mensaje que saldra en la ventana osea adentro pue
        JLabel SaludoMesj = new JLabel("Bienvenido a POKEMON STADIUM TEXT VERSION", SwingConstants.CENTER);
        SaludoMesj.setFont(new Font("Papyrus", Font.BOLD, 19));

        //crearemos el boton de salir y se va a ordenar con el mensaje HOY SI
        JPanel bP = new JPanel();
        JButton Malir = new JButton("Salir");
        Malir.setPreferredSize(new Dimension(100, 30));
        bP.add(Malir);

        mainPanel.add(SaludoMesj);
        mainPanel.add(bP); //referencia a los simson de malir sol
        add(mainPanel);

        //Hilo para reproducir el ost
        mPlayer = new MusicPlayer(audiofilepath);
        Thread musicThread = new Thread(mPlayer);
        musicThread.start();
    }

    package practica2;

    import javax.swing.SwingUtilities;
}

/*
 * @author kiquemarroquin
 */
public class Practica2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            //Creamos una nueva instancia para la ventana de bienvenida
            Bienvenida welcomeWindow = new Bienvenida("/Users/kiquemarroquin/Desktop/10thAnniversaryStreamSoundtrack/Song_That_Plays_When_Somebody_Verses_Sans_(by_Carlos_Insaneinthe
//Cambio la cancion para que no me desesperando de oirla a CADA RATO
//que se vea
            welcomeWindow.setVisible(true);
        });
    }
}
```

Seguimos con el menú principal donde están todos los botones, que terminaron siendo, agregar personaje, modificar, eliminar, visualizar, simular combate, ver historial de combate, buscar personaje por nombre, guardar, cargar limpiar el estado del sistema, limpiar la bitácora de acciones, y ver los datos de estudiante, que esto es gracias, a que en el panel principal, se agregan los botones, gracias a la librería, JButton para los botones, JLabel para el título, JPanel para colocar una “ventana” donde todos los botones, título etc. Calcen en la ventana.

```

public menuPrincipal() {
    setTitle("Arena USAC----- Menú Principal");
    //el tamaño de la venta
    setSize(500, 600);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //se cierra la ventana la app se termina

    //un panel con un GridLayout de 10 filas y una columna (por las opciones)
    JPanel panel = new JPanel(new GridLayout(10, 1, 10, 10));

    //un titulo bonito para el menu
    JLabel titulo = new JLabel("Menu Principal", SwingConstants.CENTER);
    titulo.setFont(new Font("Arid", Font.BOLD, 25));
    panel.add(titulo);

    //creamos TODOS LOS BOTONES PARA CADA OPCIÓN
    JButton btAgregar = new JButton("1. Agregar Personaje");
    //accion del boton agregar personaje
    btAgregar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Personajes.agregarVentana = new Personajes();
            agregarVentana.setVisible(true);
        }
    });
    JButton btModificar = new JButton("2. Modificar Personaje");
    //accion del boton de modificar personajes
    btModificar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            modificarPersonaje.modificarVentana = new modificarPersonaje();
            modificarVentana.setVisible(true);
        }
    });

    JButton btHistorial = new JButton("6. Ver Historial de Batallas");
    //accion del boton de historial
    btHistorial.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            historialCombate.VentanaHistorial = new historialCombate();
            VentanaHistorial.setVisible(true);
        }
    });

    JButton btBuscar = new JButton("7. Buscar Personaje");
    //accion del boton este...
    btBuscar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            BuscarPer.VentanaBuscar = new BuscarPer();
            VentanaBuscar.setVisible(true);
        }
    });

    JButton btGuardar = new JButton("8. Guardar Estado del Sistema");
    //accion del boton guardar
    btGuardar.addActionListener(e -> gestorArchivo.GuardarPersonaje());

    JButton btCargar = new JButton("9. Cargar Estado del sistema");
    //accion del cargar
    btCargar.addActionListener(e -> gestorArchivo.CargarPersonajes());

    JButton btLimpiar = new JButton("10. Limpiar Estado del Sistema");
    //accion del boton de limpiar
    btLimpiar.addActionListener(e -> gestorArchivo.BorrarDatos());

    JButton btBitcora = new JButton("11. Limpiar bitacora de acciones");
    //accion boton de bitacora,
    btBitcora.addActionListener(e -> validarAcción.MostrarBit());
}

public btEliminar() {
    //Acción del botón 3
    btEliminar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            eliminarPersonaje.eliminarVentana = new eliminarPersonaje();
            eliminarVentana.setVisible(true);
        }
    });
}

JButton btVer = new JButton("4. Ver Personajes Registrados");
//Acción del botón de ver personajes
btVer.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        verPersonaje.verVentana = new verPersonaje();
        verVentana.setVisible(true);
    }
});

JButton btSimular = new JButton("5. Simulación de Combates");
//Acción botón de MAdrizes
btSimular.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SimPelea.combateVentana = new SimPelea();
        combateVentana.setVisible(true);
    }
});

JButton btYO = new JButton("12. Ver Datos del Estudiante ");
btYO.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        YOOOOO.ventanaYO = new YOOOOO(); // si soy yo, que mas xd
        ventanaYO.setVisible(true);
    }
});

//en mac el menu se ve bonito la verdad, VIVA MAC y sus precios no
panel.add(btAgregar);
panel.add(btModificar);
panel.add(btEliminar);
panel.add(btVer);
panel.add(btSimular);
panel.add(btHistorial);
panel.add(btBuscar);
panel.add(btGuardar);
panel.add(btCargar);
panel.add(btLimpiar);
panel.add(btBitcora);
panel.add(btYO);
add(panel);
}
}

```

Seguimos con la primera opción de agregar personajes, creamos variables públicas que podamos hacer llamado desde cualquier otra parte, que ya verán que eso será útil en el futuro, siguiendo con la clase, con el JPanel, creamos una ventana para que el usuario ingrese los valores que indica cada apartado, sea el nombre, puntos de vida, puntos de ataque, velocidad, etc. Siguiendo en la clase de personajes, creamos una clase para guardar los personajes en la matriz de personajes (de máximo 100 personajes), que se hace llamado de esta en la acción del botón de “guardar”, se creamos las validaciones para cada apartado numérico que hay que ingresar, por si el usuario ingresa un valor que no está permitido, o si no llena todos los apartados, de allí al llenarlos todos, los datos ingresados se guardan en la matriz de personajes, al ingresar un personaje se puede ingresar cuantos se quiera, el id el programa lo asigna automáticamente, y todo con sus validaciones para la bitácora de acciones que a continuación se habla.

```
* @author kiquemarroquin
*/
public class Personajes extends JFrame {
    //una matriz de 50*9 para guardar a los personajes
    public static String[][] personaje = new String[100][9];

    //Un contador estatico para que generar el ID sea unico y poderlos rastrear
    public static int CantPersonajes = 0;
    public static int UltID = 0;
    public static int contadorID = 1;
    // contantes para la matriz
    public static final int ID = 0;
    public static final int NOMBRE = 1;
    public static final int ARMA = 2;
    public static final int HP = 3;
    public static final int ATAQUE = 4;
    public static final int VELOCIDAD = 5;
    public static final int AGILIDAD = 6;
    public static final int DEFENSA = 7;
    public static final int VICTERR = 8;//Victorias y derrotas
    public static final int NUMCAMPOSPERSONAJE = 9;

    //componentes de esta interfaz
    private JTextField txtNombre;
    private JTextField txtArma;
    private JTextField txtHp;
    private JTextField txtAtaque;
    private JTextField txtVelocidad;
    private JTextField txtAgilidad;
    private JTextField txtDefensa;

    //una clase publica para hacer llamados en todas partes mejor, ya me irritede hacer lo mismo ashuddaaaa
    public static int buscarPer(String busqueda){
        if (busqueda == null || busqueda.trim().isEmpty()){
            return -1;
        }
        String buscar = busqueda.trim().toLowerCase();
        for (int i = 0; i < CantPersonajes; i++){
            String id = personaje [i][ID];
            String nombre = personaje [i][NOMBRE ].toLowerCase();

            if (id.equals(buscar) || nombre.equals(buscar)){
                return i; //devolveremos el indice del personaje que encontramos
            }
        }
        return -1;
    }

    //ahora si la clase para guardar el personaje
    public Personajes(){
        setTitle("Agregar Personaje");
        setSize(900, 900);
        setLocationRelativeTo(null); //para CENTRARLA
        //solo se va a ocultar para poder reusarlo, viva el recicaje
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JPanel panel = new JPanel(new GridLayout(9, 2, 10, 10));
        //el titulo del panel NO ARRIBITA si no que... bueno DENTRO del panel
        JLabel titulo = new JLabel("Agregue un nuevo personaje", SwingConstants.CENTER);
        titulo.setFont(new Font("Arial", Font.BOLD, 18));
        panel.add(titulo);
        panel.add(new JLabel(" ")); //un espacio en blanco como el System.out.println();
    }
}
```

```

// los apartados que se deben de ser llenados para agregar un personaje
panel.add(new JLabel("Escribe el nombre que quieras:"));
txtNombre = new JTextField();
panel.add(txtNombre);

panel.add(new JLabel("Elige el arma de tu preferencia:"));
txtArma = new JTextField();
panel.add(txtArma);

panel.add(new JLabel("Escribe los puntos de vida del personaje (100 o 500):"));
txtHp = new JTextField();
panel.add(txtHp);

panel.add(new JLabel("Escribe los puntos de ataque del personaje (10 a 100):"));
txtAtaque = new JTextField();
panel.add(txtAtaque);

panel.add(new JLabel("Inserta la velocidad del personaje (1 a 10):"));
txtVelocidad = new JTextField();
panel.add(txtVelocidad);

panel.add(new JLabel("Escribe la agilidad del personaje(1 a 10):"));
txtAgilidad = new JTextField();
panel.add(txtAgilidad);

panel.add(new JLabel("Escribe la defensa que tendra el personaje (1 a 50):"));
txtDefensa = new JTextField();
panel.add(txtDefensa);

// un boton para guardar los datos
JButton btGuardar = new JButton("Guardar cambios");
panel.add(btGuardar);

add(panel);

btGuardar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        guardarPersonaje();
    }
});
}

private void guardarPersonaje() {
    if (CantPersonajes > 100) {
        JOptionPane.showMessageDialog(this, "El limite de personajes se alcanzo, borra alguno.", "Error 001", JOptionPane.ERROR_MESSAGE);
        validarAccion.regisAcción("agregar personaje", false, "Salio mal");
        return;
    }
    // todo esto es para que no hayan espacios sin llenar :3 may intelliget
    String nombre = txtNombre.getText();
    if (nombre.isEmpty() || txtArma.getText().isEmpty() || txtHp.getText().isEmpty() || txtAtaque.getText().isEmpty() || txtVelocidad.getText().isEmpty() || txtAgilidad.getText().isEmpty() || txtDefensa.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Por favor llena todos los apartados D:", "Error 002", JOptionPane.ERROR_MESSAGE);
        validarAccion.regisAcción("agregar personaje", false, "Salio mal APROPOSITO");
        return;
    }

    int hp, ataque, velocidad, agilidad, defensa;
    try {
        hp = Integer.parseInt(txtHp.getText());
        if (hp < 10 || hp > 500) {
            JOptionPane.showMessageDialog(this, "Los puntos de vida deben de ser un numero entero de 100 o 500.", "Error 003", JOptionPane.ERROR_MESSAGE);
            validarAccion.regisAcción("agregar vida personaje", false, "Salio mal APROPOSITO");
            return;
        }

        ataque = Integer.parseInt(txtAtaque.getText());
        if (ataque < 10 || ataque > 100) {
            JOptionPane.showMessageDialog(this, "Los puntos de ataque deben de ser un numero entero de 10 a 100.", "Error 004", JOptionPane.ERROR_MESSAGE);
            validarAccion.regisAcción("agregar ataque personaje", false, "Salio mal APROPOSITO");
            return;
        }

        velocidad = Integer.parseInt(txtVelocidad.getText());
        if (velocidad < 1 || velocidad > 10) {
            JOptionPane.showMessageDialog(this, "La velocidad debe de ser un numero entero de 1 a 10.", "Error 005", JOptionPane.ERROR_MESSAGE);
            validarAccion.regisAcción("agregar velocidad personaje", false, "Salio mal APROPOSITO");
            return;
        }

        agilidad = Integer.parseInt(txtAgilidad.getText());
        if (agilidad < 1 || agilidad > 10) {
            JOptionPane.showMessageDialog(this, "La agilidad debe de ser un numero entero de 1 a 10.", "Error 006", JOptionPane.ERROR_MESSAGE);
            validarAccion.regisAcción("agregar agilidad personaje", false, "Salio mal APROPOSITO");
            return;
        }

        defensa = Integer.parseInt(txtDefensa.getText());
        if (defensa < 1 || defensa > 50) {
            JOptionPane.showMessageDialog(this, "La defensa debe de ser un numero entero de 1 a 50.", "Error 007", JOptionPane.ERROR_MESSAGE);
            validarAccion.regisAcción("agregar defensa personaje", false, "Salio mal APROPOSITO");
            return;
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Los apartados que son numeros deben ser llenados con eso... pos numeros", "Error 008", JOptionPane.ERROR_MESSAGE);
        validarAccion.regisAcción("agregar personaje", false, "Salio mal APROPOSITO");
        return;
    }

    personaje[CantPersonajes][ID] = String.valueOf(Personajes.contadorID);
    personaje[CantPersonajes][NOMBRE] = txtNombre.getText();
    personaje[CantPersonajes][ARMA] = txtArma.getText();
    personaje[CantPersonajes][HP] = txtHp.getText();
    personaje[CantPersonajes][ATAQUE] = txtAtaque.getText();
    personaje[CantPersonajes][VELOCIDAD] = txtVelocidad.getText();
    personaje[CantPersonajes][AGILIDAD] = txtAgilidad.getText();
    personaje[CantPersonajes][DEFENSA] = txtDefensa.getText();
    personaje[CantPersonajes][VICDER] = "0-0"; //Inciamos esto con nada

    CantPersonajes++;
    contadorID++;

    JOptionPane.showMessageDialog(this, txtNombre.getText() + " se ha agregado al roster de personajes");
    txtNombre.setText("");
    txtArma.setText("");
    txtHp.setText("");
    txtAtaque.setText("");
    txtVelocidad.setText("");
    txtAgilidad.setText("");
    txtDefensa.setText("");
    validarAccion.regisAcción("agregar personaje", true, "Salio bien");
}
}

```

Siguiendo con el el apartado de modificar personaje creamos apartados privados de texto correspondientes al nombre, arma, hp, ataque, etc, y los apartados de areas de texto a los apartados que son modificables, que son el arma, hp, ataque, velocidad, agilidad, defensa, y el boton para guardar cambios, creamos una ventana para el apartado de modificar personaje, con la opcion de “*DISPOSE_ON_CLOSE*”, que al cerrarse la ventana se cierra el proceso, agregamos el titulo, y el area de

texto para buscar el personaje por ID o por nombre, agregamos un boton para ejecutar la accion de buscar, y los apartados donde nos indica cuales son los atributos actuales del personaje encontrado, y al ser encontrado se podra modificar los atributos de este mismo, y el boton de guardar los cambios, los datos actuales del personaje a encontrar se muestran gracias al metodo de mostrar personajes que muestra tanto los datos actuales como los modificados despues de guardar, siguiendo con esa palabra, al guardar los datos, se tiene las validaciones de cada valor numerico, se actualiza la matriz y con el ultimo metodo se encarga de limpiar los apartados luego de guardar los datos.

```

JPanel panel = new JPanel(new GridLayout(12, 2, 10, 20));
//titulo de la ventana
JLabel titulo = new JLabel("Modificar personaje", SwingConstants.CENTER);
titulo.setFont(new Font("Arial", Font.BOLD, 20));
panel.add(titulo);
panel.add(new JLabel(""));

//Apartado para buscar el personaje
panel.add(new JLabel("Buscar por ID o Nombre"));
txtBuscar = new JTextField();
panel.add(txtBuscar);

btBuscar = new JButton("Buscar");
panel.add(btBuscar);
panel.add(new JLabel(""));

//Esto es para que se vean los datos actuales
LbNombreAct = new JLabel("Nombre: ");
panel.add(LbNombreAct);
LbArmaAct = new JLabel("Arma: ");
panel.add(LbArmaAct);
LbHPAct = new JLabel("Punto de salud: ");
panel.add(LbHPAct);
LbAtaqueAct = new JLabel("Ataque: ");
panel.add(LbAtaqueAct);
LbVelocidadAct = new JLabel("Velocidad: ");
panel.add(LbVelocidadAct);
LbAgilidadAct = new JLabel("Agilidad: ");
panel.add(LbAgilidadAct);
LbDefensaAct = new JLabel("Defensa: ");
panel.add(LbDefensaAct);
panel.add(new JLabel(""));
panel.add(new JLabel(""));

public modificarPersonaje() {
    setTitle("Modificar Personaje");// titulo de la ventana
    setSize(500, 600);// tamaño de la ventana
    setLocationRelativeTo(null);// centramos la ventana
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); //ya se la saben para que al cerrar se cierre el proceso
}

```

```

//Campos que se van a llenar como los nuevos datos
panel.add(new JLabel("Escoge el arma de tu preferencia:"));
txtNArma = new JTextField();
panel.add(txtNArma);

panel.add(new JLabel("Ingresa los puntos de vida (100 a 500):"));
txtNHP = new JTextField();
panel.add(txtNHP);

panel.add(new JLabel("Ingresa los puntos de ataque (10 a 100):"));
txtNAtaque = new JTextField();
panel.add(txtNAtaque);

panel.add(new JLabel("Ingresa los puntos de velocidad (1 a 10):"));
txtNVelocidad = new JTextField();
panel.add(txtNVelocidad);

panel.add(new JLabel("Ingresa los puntos de agilidad (1 a 10):"));
txtNAgilidad = new JTextField();
panel.add(txtNAgilidad);

panel.add(new JLabel("Ingresa los puntos de defensa:"));
txtNDefensa = new JTextField();
panel.add(txtNDefensa);

btGuardar = new JButton("Guardar Cambios");
panel.add(btGuardar);

//Agregamos todo esto a la ventana en cuestion
add(panel);

btBuscar.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e) {
        buscarPer();
    }
});
}

//un metodo para mostrar el personaje, solo que aca es con JLabel en vez del System.out.println
private void mostrarPer(int i) {
    String[] datos = Personajes.personaje[i];
    LbNombreAct.setText("Nombre: " + datos[Personajes.NOMBRE]);
    LbArmaAct.setText("Arma: " + datos[Personajes.ARMA]);
    LbHPAct.setText("Puntos de vida: " + datos[Personajes.HP]);
    LbAtaqueAct.setText("Ataque: " + datos[Personajes.ATAQUE]);
    LbVelocidadAct.setText("Velocidad: " + datos[Personajes.VELOCIDAD]);
    LbAgilidadAct.setText("Agilidad: " + datos[Personajes.AGILIDAD]);
    LbDefensaAct.setText("Defensa: " + datos[Personajes.DEFENSA]);

    //Cargamos los datos para editar para que sea mas facil modificarlos
    txtNArma.setText(datos[Personajes.ARMA]);
    txtNHP.setText(datos[Personajes.HP]);
    txtNAtaque.setText(datos[Personajes.ATAQUE]);
    txtNVelocidad.setText(datos[Personajes.VELOCIDAD]);
    txtNAgilidad.setText(datos[Personajes.AGILIDAD]);
    txtNDefensa.setText(datos[Personajes.DEFENSA]);
}

private void guardarDatos() {
    if (tfEncuentro == -1) {
        JOptionPane.showMessageDialog(this, "Por favor, buscar un personaje primero", "Error 013", JOptionPane.ERROR_MESSAGE);
        validarAcción.regisAcción("Buscar personaje para modificarlo", false, "Salio MAL APROPOSITO");
        return;
    }

    //Validar los valores:
    try {
        int hp = Integer.parseInt(txtNHP.getText());
        if (hp < 100 || hp > 500) {
            JOptionPane.showMessageDialog(this, "Los puntos de vida deben de estar entre 100 y 500", "Error 014", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción("Modificar vida personaje", false, "Salio MAL APROPOSITO");
            return;
        }

        int ataque = Integer.parseInt(txtNAtaque.getText());
        if (ataque < 10 || ataque > 100) {
            JOptionPane.showMessageDialog(this, "Los puntos de ataque deben de estar entre 10 y 100", "Error 015", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción("Modificar ataque personaje", false, "Salio MAL APROPOSITO");
            return;
        }

        int velocidad = Integer.parseInt(txtNVelocidad.getText());
        if (velocidad < 1 || velocidad > 10) {
            JOptionPane.showMessageDialog(this, "La velocidad debe de estar entre 1 a 10", "Error 016", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción("Modificar velocidad personaje", false, "Salio MAL APROPOSITO");
            return;
        }

        int agilidad = Integer.parseInt(txtNAgilidad.getText());
        if (agilidad < 1 || agilidad > 10) {
            JOptionPane.showMessageDialog(this, "La agilidad debe de estar entre 1 a 10", "Error 017", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción("Modificar agilidad personaje", false, "Salio MAL APROPOSITO");
            return;
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Los apartados que son numeros deben ser llenados con eso... por numeros", "Error 018", JOptionPane.ERROR_MESSAGE);
        validarAcción.regisAcción("Modificar personaje", false, "Salio MAL APROPOSITO");
    }

    //le agregamos la accion al boton de guardar
    btGuardar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            guardarDatos(); //Un llamado a un metodo
        }
    });

    private void buscarPer() {
        String buscar = txtBuscar.getText().trim();
        pEncontrar = -1; //resetear el indice

        if (buscar.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Ingresa un ID o Nombre para buscar", "Error 011", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción("Buscar personaje para modificarlo", false, "Salio MAL APROPOSITO");
            return;
        }

        //el ciclo for de toda la vida para buscar la cadena de texto en la matriz de personaje
        for (int i = 0; i < Personajes.CANTPERSONAJES; i++) {
            String Id = Personajes.personaje[i][Personajes.ID];
            String nombre = Personajes.personaje[i][Personajes.NOMBRE];

            if (Id.equals(buscar) || nombre.equalsIgnoreCase(buscar)) {
                pEncontrar = i;
                mostrarPer(i); //Otro llamado
                return;
            }
        }

        JOptionPane.showMessageDialog(this, "El personaje no se ha encontrado", "Error 012", JOptionPane.ERROR_MESSAGE);
        validarAcción.regisAcción("Buscar personaje para modificarlo", false, "Salio MAL");
    }
}

```

Continuando con eliminar personajes, igual creamos una ventana con su titulo, y su boton para buscar, y su area de texto para escribir el nombre del personaje para eliminarlo, la accion del boton para eliminar el personaje, para esto ultimo se creo un metodo para eso mismo, se hace el llamado de un metodo creado para buscar personajes por id y nombre, si se encuentra, salta una ventana de confirmacion para eliminar el personaje, con la opcion de si se borra los datos del personaje, pero si este ya peleo con otro personaje en el historial de combates sale solo su nombre, por lo que sus datos “no existen” solo se cambia los valores como si no existe, con sus validaciones para la bitacora de acciones.

```

public eliminarPersonaje(){
    setTitle("Eliminar Personaje");
    setSize(700, 600);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JPanel panel = new JPanel(new GridLayout(4, 1, 10, 20));

    JLabel titulo = new JLabel("Eliminar Personaje", SwingConstants.CENTER);
    titulo.setFont(new Font("Arial", Font.BOLD, 18));
    panel.add(titulo);

    panel.add(new JLabel("Escribe ACA el ID o el nombre del personaje que deseas eliminar"));
    txtBuscar = new JTextField();
    panel.add(txtBuscar);

    JPanel pBoton = new JPanel(new FlowLayout());
    btBuscar = new JButton("Buscar");
    pBoton.add(btBuscar);
    panel.add(pBoton);

    LbEncuentrado = new JLabel("Tamus buscando", SwingConstants.CENTER);
    LbEncuentrado.setVisible(false);
    panel.add(LbEncuentrado);

    btEliminar = new JButton("Eliminar Personaje");
    btEliminar.setVisible(false);
    panel.add(btEliminar);
    //añadimos todo eso de arriba al panel o como le dire de ahora en adelante, plantilla
    add(panel);

    btBuscar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            PersonajeEl();
        }
    });

    btEliminar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            confirmarEliminar();
        }
    });
}

private void PersonajeEl() {
    String Buscar = txtBuscar.getText().trim();
    Pecontrado = -1;
    LbEncuentrado.setVisible(false);
    btEliminar.setVisible(false);

    if (Buscar.isEmpty()) {
        JOptionPane.showMessageDialog(this, "INGRESA EL NOMBRE O EL ID CTM", "Error 020", JOptionPane.ERROR_MESSAGE);
        validarAcción.regisAcción("buscar al personaje pa eliminarlo", false, "Salio MAL APROPOSITO");
        return;
    }

    Pecontrado = Personajes.buscarPer(Buscar);

    if (Pecontrado != -1) {
        String Neccontrado = Personajes.personaje[Pecontrado][Personajes.NOMBRE];
        LbEncuentrado.setText("El personaje a eliminar es: " + Neccontrado);
        LbEncuentrado.setVisible(true);
        btEliminar.setVisible(true);
        validarAcción.regisAcción("encontrar el personaje para eliminarlo", true, "Salio bien");
    }

    JOptionPane.showMessageDialog(this, "No se ha encontrado al personaje, f por ti", "Error 021", JOptionPane.ERROR_MESSAGE);
    validarAcción.regisAcción("Buscar al personaje para borrarlo", false, "Salio MAL");
}

```

```

private void confirmarEliminar() {
    if (Pcontrado != null) {
        String nEliminar = Personajes.personaje[Pcontrado][Personajes.NOMBRE];
        int Confirmar = JOptionPane.showConfirmDialog(
            this,
            "Estás seguro de eliminar a: " + nEliminar + "?",
            "Confirmar Eliminación",
            JOptionPane.YES_NO_OPTION);
        if (Confirmar == JOptionPane.YES_OPTION) {
            EliminaPersonaje(Pcontrado);
            JOptionPane.showMessageDialog(this, nEliminar + " se ha eliminado del roster de personajes");
            dispose();
            validarAcción.regisAcción("Borrar el personaje", true, "Salio bien");
        } else {
            validarAcción.regisAcción("Eliminar el personaje", true, "Salio... y ya");
        }
    }
}

private void EliminarPersonaje(int in) {
    //Obtenemos el historial de peleas... si es que hay
    String historial = Personajes.personaje[in][Personajes.VICDER];
    boolean GP = historial != null && !historial.isEmpty() && historial.equals("0-0");

    if (GP) {
        //si el personaje no se ha peleado con nadie, vamos a decir que no hay nada, como el meme de avatar
        Personajes.personaje[in][Personajes.NOMBRE] = "Ya no esta";
        Personajes.personaje[in][Personajes.ARMA] = "No hay";
        Personajes.personaje[in][Personajes.HP] = "No hay";
        Personajes.personaje[in][Personajes.ATAQUE] = "No hay";
        Personajes.personaje[in][Personajes.VELOCIDAD] = "No hay";
        Personajes.personaje[in][Personajes.AGILIDAD] = "No hay";
        Personajes.personaje[in][Personajes.DEFENSA] = "No hay";
    } else {
        // si el personaje no se ha sacado la chucha con alguien mas lo eliminamos totalmente y ya
        for (int i = in; i < Personajes.CantPersonajes - 1; i++) {
            Personajes.personaje[i] = Personajes.personaje[i + 1];
        }
        Personajes.personaje[Personajes.CantPersonajes - 1] = new String[9];
        Personajes.CantPersonajes--;
    }
}

```

Seguimos con la opcion de de visualizar personajes, en este tambien se hace la ventana que contendra el titulo, y una tabla que contiene los datos de los personajes que ya fueron ingresados, con los datos tenemos, el id, el nombre, el arma, los puntos de vida, la velocidad, la agilidad, la defensa y el historial de victorias y derrotas. Esta tabla se configura con el JscrollPane para que se vaya desplazando automaticamente, con la info de los personajes, y Jtable que es la tabla que los contendra, en las columnas.

```

/*
public class verPersonaje extends JFrame {
    private JTable tablaPerson;
    private JScrollPane panelS;

    public verPersonaje(){
        //titulo de la ventana
        setTitle("Personajes Registrados");
        setSize(800, 800);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        //nombres de las columnas
        String[] ncolumns = {"ID", "Nombre", "Arma", "Hp", "Ataque", "Velocidad", "Agilidad", "Defensa", "Historial"};

        //para arreglar las filas usamos defaultable Model, esto lo encontre en satackoverflow :
        DefaultTableModel modTabla = new DefaultTableModel(ncolumns, 0);
        tablaPerson = new JTable(modTabla);

        //llenamos la tabla con los personajes del metodo que esta alli abajo
        TablaPersonajes(modTabla);

        panelS = new JScrollPane(tablaPerson);
        add(panelS, BorderLayout.CENTER );
    }

    private void TablaPersonajes(DefaultTableModel mod){
        //vamos a limpiar las filas que ya esten, cuando se agreguen o borrar personajes
        mod.setRowCount(0);

        //un hermoso ciclo for para recorrer la matriz de personajes y agregar cada fila a la tabla de personajes
        for(int i=0; i < Personajes.CantPersonajes; i++){
            //Creamos un arreglo con los datos de cada personaje en la posicion de i
            Object[] filas = new Object[Personajes.personaje[i].length];
            for(int j = 0; j < Personajes.personaje[i].length; j++){
                filas[j] = Personajes.personaje[i][j];
            }
            mod.addRow(filas);
        }
    }
}

```

Seguimos con la parte mas dificil de la practica que es la de simular personajes, aca la ventana contiene, los JComboBox para el luchador 1 como para el luchador 2, junto al boton para que inicie el combate, a la hora de empezar el combate, se hace llamado al metodo de iniciar combate, dicho metodo hace llamado la matriz de personajes junto al nombre del personaje y lo tratamos como un objeto haciendo las validaciones correspondientes, tenemos un metodo antes que es para cargar el personaje, haciendo llamado a un ciclo for que busque ambos personajes en la matriz, se inicia el hilo correspondiente al luchador 1 como para el luchador 2, a la hora del combate se hace un bucle while para que se termine cuando el luchador 1 como el luchador 2, se quede a 0 o menos puntos de vida, se tiene un contador por cada turno pasado, se hace un calculo de daño que toma el daño del luchador 1 contra la defensa del luchador 2, esto teniendo en cuenta el thread.sleep que toma la velocidad del personaje dividido 1000, y la velocidad del personaje junto a la librería random que es una “probabilidad” usa la agilidad del personaje, esto con la condicion que si el luchador 1 o el luchador 2 queda a 0 de vida antes este sera el perdedor y por contraparte el otro sera el ganador, seguimos con el ultimo metodo de la clase, que se encarga de registrar al perdedor y ganador del combate, junto a su contador que esta a 0 para iniciar, se hace llamado a la matriz para el apartado de victorias y derrotas, para luego aparecer en pantalla, el conteo de victorias y derrotas, junto a los turnos, la hora y fecha.

```

public class SimPelea extends JFrame {
    //esto es para buscar los personajes de la matriz

    private JComboBox<String> cbLuchador1;
    private JComboBox<String> cbLuchador2;

    private JButton btIniLucha;
    private JTextArea bitacoraArea;
    private JScrollPane sP;

    private Random random = new Random();

    public SimPelea() {
        //esto ya se lo saben
        setTitle("Simulador de Combates");
        setSize(700, 600);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        //otro chupas para que el texto si se ve...
        JPanel PanelSuperior = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JPanel panelPrincipal = new JPanel(new BorderLayout(10, 10));
        JLabel titulo = new JLabel("Elige dos personajes para LUCHAR", SwingConstants.CENTER);
        titulo.setFont(new Font("Arial", Font.BOLD, 18));

        cbLuchador1 = new JComboBox<String>();
        cbLuchador2 = new JComboBox<String>();
        CargarLuchador(); //Un metodo que alli abajo esta xd

        PanelSuperior.add(titulo);
        PanelSuperior.add(new JLabel("Elige al primer luchador"));
        PanelSuperior.add(cbLuchador1);
        PanelSuperior.add(new JLabel("Ahora Elige al segundo luchador"));
        PanelSuperior.add(cbLuchador2);

        btIniLucha = new JButton("MADRAZOS");
        PanelSuperior.add(btIniLucha);

        //otro chupas para evitar que el boton crashee el programa
        bitacoraArea = new JTextArea();
        bitacoraArea.setEditable(false);
        JScrollPane sP = new JScrollPane(bitacoraArea);
    }
}

```

```

//agregar los componentes a la region del panel principal
panelPrincipal.add(panelSuperior, BorderLayout.NORTH); //el panel con el titulo, el botn y el seleccionador
panelPrincipal.add(sP, BorderLayout.CENTER);
add(panelPrincipal);

btIniLucha.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        IniciarCombate(); //();//metodo que alli ta MAS abajo
    }
});

private void CargarLuchador() {
    for (int i = 0; i < Personajes.CantPersonajes; i++) {
        String nombre = Personajes.personajes[i].getNombre();
        cbLuchador1.addItem(nombre);
        cbLuchador2.addItem(nombre);
    }
}

```

```

private void LuchaCombate(){
    String Luchador1 = (String) cbLuchador1.getSelectedItem();
    String Luchador2 = (String) cbLuchador2.getSelectedItem();

    //otro chapas...
    if (Personajes.getPersonaje > 0) {
        JOptionPane.showMessageDialog(this, "NECESITAS ALMENOS DOS Personajes para el combate", "Error 022", JOptionPane.ERROR_MESSAGE);
        validarAcciones.regisAcción("Madrazos entre personajes", false, "Salio MAL APROPOSITO");
    }
    return;
}

if (Luchador1 == null || Luchador2 == null) {
    JOptionPane.showMessageDialog(this, "Selecciona DOS Personajes para el combate", "Error 023", JOptionPane.ERROR_MESSAGE);
    validarAcciones.regisAcción("Madrazos entre personajes", false, "Salio MAL APROPOSITO");
}

if (Luchador1.equals(Luchador2)) {
    JOptionPane.showMessageDialog(this, "Un personaje no puede pelear solito, serio muy raro eso", "Error 024", JOptionPane.ERROR_MESSAGE);
    validarAcciones.regisAcción("Madrazos entre personajes", false, "Salio MAL APROPOSITO");
}

int indice1 = Personajes.buscarPorNombre(Luchador1);
int indice2 = Personajes.buscarPorNombre(Luchador2);

if (indice1 < -1 || indice2 < -1) {
    JOptionPane.showMessageDialog(this, "Algun personaje, no existe, ¿Qué? no se la verdad", "Error 025", JOptionPane.ERROR_MESSAGE);
    validarAcciones.regisAcción("Madrazos entre personajes", false, "Salio MAL APROPOSITO");
}

//los luchadores
String[] L1 = Personajes.personajes[ indice1 ];
String[] L2 = Personajes.personajes[ indice2 ];

//los puntos de vida de cada uno
int Hp1 = Integer.parseInt(L1[Personajes.HP]);
int Hp2 = Integer.parseInt(L2[Personajes.HP]);

if (Hp1 < 0 || Hp2 < 0) {
    JOptionPane.showMessageDialog(this, "Alguno de los personajes ya se maria xD", "Error 026", JOptionPane.ERROR_MESSAGE);
    validarAcciones.regisAcción("Madrazos entre personajes", false, "Salio MAL");
    return;
}

private void Combate(String atacante, String defensor, JTextArea bitacora) {
    String Natacante = atacante[Personajes.NOMBRE];
    String Ndefensor = defensor[Personajes.NOMBRE];
    int turnos = 0;
    while (Integer.parseInt(atacante[Personajes.HP]) > 0 && Integer.parseInt(defensor[Personajes.HP]) > 0) {
        try {
            turnos++;
            System.out.println("Turno: " + turnos);
            int dañoBase = Integer.parseInt(atacante[Personajes.ATAQUE]);
            int agilDef = Integer.parseInt(defensor[Personajes.AGILIDAD]);
            int defDef = Integer.parseInt(defensor[Personajes.DEFENSA]);

            //la probabilidad de esquivar
            boolean esquivar = random.nextInt(10) > agilDef;

            if (esquivar) {
                int t = turnos;
                SwingUtilities.invokeLater(() -> bitacora.append("\nTurno" + t + ":" + Natacante + " HA ATACADO, PERO \n" + Ndefensor + " HA ESQUIVADO EL ATAQUE"));
            } else {
                int dañoFinal = dañoBase - defDef;
                if (dañoFinal < 0) {
                    dañoFinal = 0;
                }
                int hpActualDef = Integer.parseInt(defensor[Personajes.HP]);
                hpActualDef -= dañoFinal;
                defensor[Personajes.HP] = String.valueOf(hpActualDef);

                //Quiero para evitar un error
                int t2 = turnos;
                final int dañoFinal;
                final int HpActualDef = hpActualDef;
                if (t2 < turnos) {
                    System.out.println("Error en el daño final");
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    //Es la linea mas larga que he echo... dist... me habrá ganado un record guiness
    SwingUtilities.invokeLater(() -> bitacora.append("\nTurno" + t + ":" + Natacante + " HA ECHO UN ATAQUE " + Ndefensor + " HA SUFERIDO \n" + dañoFinal + " PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE " + Ndefensor + " Es: " + HpActualDef));
}

int veloAtacante = Integer.parseInt(atacante[Personajes.VELOCIDAD]);
Thread.sleep(1000 / veloAtacante);
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
}
return;
}

private void actualHistorial(String ganador, String perdedor, int turnos) {
    //la hora y fecha
    SimpleDateFormat fechaform = new SimpleDateFormat("yyyy-MM-dd");
    SimpleDateFormat horiform = new SimpleDateFormat("HH:mm:ss");

    Date ahora = new Date();
    String fecha = fechaform.format(ahora);
    String hora = horiform.format(ahora);

    //otro chapas
    int vicAct = 0;
    int derrAct = 0;

    int inGanar = Personajes.buscarPorNombre(ganador);
    int inPerder = Personajes.buscarPorNombre(perdedor);

    String HistGanar = Personajes.personaje[inGanar][Personajes.VICDERR];
    String HistPer = Personajes.personaje[inPerder][Personajes.VICDERR];

    //sin esto el programa falla... y tuve que rehacer esta basura
    try {
        String[] datos = HistGanar.split("\\|");
        vicAct = Integer.parseInt(datos[0].trim());
        derrAct = Integer.parseInt(datos[1].trim());
    } catch (Exception e1) {
        try {
            String[] datos = HistGanar.split(":");
            vicAct = Integer.parseInt(datos[0].trim());
            derrAct = Integer.parseInt(datos[1].trim());
        } catch (Exception e2) {
            vicAct = 0;
            derrAct = 0;
        }
    }
    vicAct++;
    Personajes.personaje[inGanar][Personajes.VICDERR] = vicAct + " | " + derrAct + " | " + turnos + ":" + fecha + " | " + hora;
    //resetteamos el contenido de perdidas y ganancias
    vicAct = 0;
    derrAct = 0;
}

private void combate() {
    btInLucha.setEnable(true);
    bitacoraArea.setText("BIENVENIDOS AL COMBATE, HOY SE ENFRETARÁ " + Luchador1 + " CONTRA " + Luchador2 + " SIGAN VIENDO");
    btInLucha.setEnabled(false);

    String[] P1 = L1.clone();
    String[] P2 = L2.clone();

    //en punto de los hilos... me wo a morir
    Thread hilo1 = new Thread() -> Combate(P1, P2, bitacoraArea);
    Thread hilo2 = new Thread() -> Combate(P1, P2, bitacoraArea);
    hilo1.start();
    hilo2.start();
}

int turnosF = turnos;
//vamos arreglar esto...
SwingUtilities.invokeLater(() -> {
    String ganador = "";
    String perdedor = "";

    if (Integer.parseInt(atacante[Personajes.HP]) > 0) {
        ganador = Natacante;
        perdedor = Ndefensor;
    } else if (Integer.parseInt(defensor[Personajes.HP]) > 0) {
        ganador = Ndefensor;
        perdedor = Natacante;
    } else if (ganador.isEmpty()) {
        bitacora.append("\n" + ganador + " Ha ganado el combate en \n" + turnosF + " turnos \n");
        actualHistorial(ganador, perdedor, turnosF);
        validarAcciones.regisAcción("madrazos entre personajes", true, "Salio bien");
    } else {
        bitacora.append("TENEMOS UN EMPATE A LOS: " + turnosF + " TURNOS \n");
    }
    btInLucha.setEnabled(true);
});
}
}

try {
    String[] datos = HistPer.split("\\|");
    vicAct = Integer.parseInt(datos[0].trim());
    derrAct = Integer.parseInt(datos[1].toLowerCase());
} catch (Exception e) {
    try {
        String[] datos = HistPer.split(":");
        vicAct = Integer.parseInt(datos[0].trim());
        derrAct = Integer.parseInt(datos[1].trim());
    } catch (Exception e2) {
        vicAct = 0;
        derrAct = 0;
    }
}
derrAct++;
Personajes.personaje[inPerder][Personajes.VICDERR] = vicAct + " | " + derrAct + " | " + turnos + ":" + fecha + " | " + hora;
}
}

```

Seguimos con la parte del historial de combates, junto a esta venta se encuentra la ventana con el nombre de la misma, la fecha y hora en cual se actualiza el historial, junto a su boton del mismo, con la pantalla desplazable por el Jscrollpane, y tenemos el boton de actualizar con el llamado del metodo de cargar historial, dicho metodo se encarga de mostrar los datos del ganador, el perdedor, el ultimo combate, el numero de turnos, junto a su hora y fecha, con sus saltos de linea.

```

public historialCombate() {
    //ya se la saben... el titulo de la ventana y el tamaño de la misma... quiero dormir...
    setTitle("Historial de Combates");
    setSize(600, 700);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JPanel panelPrincipal = new JPanel(new BorderLayout(10, 10));

    JLabel titulo = new JLabel("Este es el historial de combates realizados");
    titulo.setFont(new Font("Arial", Font.BOLD, 18));
    panelPrincipal.add(titulo, BorderLayout.NORTH); //esto va a estar ARRIBA

    //agregamos el panel a la fecha y hora
    JPanel panelCentro = new JPanel(new BorderLayout());
    panelCentro.add(lbFecha, BorderLayout.CENTER);

    //Iniciamos la etiqueta de la fecha
    lbFecha = new JLabel("", SwingConstants.CENTER);
    panelCentro.add(lbFecha, BorderLayout.NORTH);

    Harea = new JTextArea();
    Harea.setEditable(false);
    JScrollPane panelAbajo = new JScrollPane(Harea);
    panelCentro.add(panelAbajo, BorderLayout.CENTER);
    panelPrincipal.add(panelCentro, BorderLayout.CENTER); //aca la kgue xd

    //el boton para que el historial se actualice
    JButton btActualizar = new JButton("Actualizar historial");
    panelPrincipal.add(btActualizar, BorderLayout.SOUTH);
    add(panelPrincipal);

    Cargarhist(); //un llamado de un metodo que esta alli abajo

    btActualizar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Cargarhist(); //la misma cosa que dije arriba.
        }
    });
}

//para que se vea mejor el historial
String victorias = datos.length > 0 ? datos[0] : "0";
String derrotas = datos.length > 1 ? datos[1] : "0";
String turnos = datos.length > 2 ? datos[2] : "N/A";
String fecha = datos.length > 3 ? datos[3] : "----";
String hora = datos.length > 4 ? datos[4] : "----";

Hist.append("Personaje: ").append(nombre).append("\n"); //salto de linea
Hist.append("Victorias: ").append(victorias).append("Derrotas: ").append(derrotas).append("\n");
Hist.append("Ultimo Combate: ").append(turnos).append(" turnos \n");
Hist.append("fecha: ").append(fecha).append(" | Hora: ").append(hora).append("\n");
Hist.append("-----\n");

validarAccion.regisAcción("ver historial de madrazos entre personajes", true, "Salio bien");

}

}
Harea.setText(Hist.toString());
}
}

private void Cargarhist() {
    SimpleDateFormat fechahora = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Date Factual = new Date();
    String HFformat = fechahora.format(Factual);

    lbFecha.setText("Ultima actualizacion: " + HFformat);

    StringBuilder Hist = new StringBuilder();

    Hist.append("Historial de Victorias y Derrotas \n");
    Hist.append("----- \n"); //es innecesario, pero da mas bonito asi

    if (Personajes.ContPersonajes == 0) {
        JOptionPane.showMessageDialog(this, "No hay personajes registrados, como para que se madrene", "Error 027", JOptionPane.ERROR_MESSAGE);
        validarAccion.regisAcción("ver historial de madrazos entre personajes", false, "Salio MAL APROPOSITO");
    } else {
        for (int i = 0; i < Personajes.ContPersonajes; i++) {
            String nombre = Personajes.personaje[i].getNombre();
            String vicDerr = Personajes.personaje[i].getVicDerr();

            String[] datos = new String[5];

            if (vicDerr.contains(",")) {
                datos = vicDerr.split(",");
            } else {
                datos[0] = "0";
                datos[1] = "0";
                datos[2] = "N/A";
                datos[3] = "----";
                datos[4] = "----";
            }

            Hist.append("Personaje: ").append(nombre).append("\n");
            Hist.append("Victorias: ").append(datos[0]).append("Derrotas: ").append(datos[1]).append("\n");
            Hist.append("Ultimo Combate: ").append(datos[2]).append(" turnos \n");
            Hist.append("fecha: ").append(datos[3]).append(" | Hora: ").append(datos[4]).append("\n");
            Hist.append("-----\n");
        }
    }
    validarAccion.regisAcción("ver historial de madrazos entre personajes", true, "Salio bien");
}
}

```

Seguimos con la opcion de buscar personaje para esta clase se crea una ventana con el nombre de la misma el área de texto para escribir el nombre del personaje que queremos buscar y un botón con el llamado al método de buscar personajes. Para el método de buscar personajes se hace llamado del método principal en la clase de personajes de buscar personajes por nombre, al encontrarse el personaje mostrará en pantalla los datos del personaje junto con el historial de victorias y derrotas haciendo llamado a la matriz de personajes al personaje asociado a la misma, luego mostrará en la en la ventana mostrará los combates el resultado del mismo los turnos y la hora en fecha en que se realizó junto a la validación de registrar accion.

```

public BuscarPer() {
    //yo no quiero explicar esta madre
    setTitle("Buscar el personaje por su nombre");
    setSize(500, 600);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JPanel PanelPrincipal = new JPanel(new BorderLayout(10, 10));

    JLabel titulo = new JLabel("Buscar Personaje", SwingConstants.CENTER);
    titulosetFont(new Font("Arial", Font.BOLD, 19));
    PanelPrincipal.add(titulo, BorderLayout.NORTH);

    //el campo para buscar
    JPanel panelBuscar = new JPanel(new GridLayout(1, 2, 5, 5));
    JTextField txtBuscar = new JTextField();
    JButton btBuscar = new JButton("Buscar Personaje");
    panelBuscar.add(new JLabel("Escribe el Personaje: "));
    panelBuscar.add(txtBuscar);

    JPanel PanelCentro = new JPanel(new BorderLayout());
    JLabel LbEncontrar = new JLabel("Información del Personaje es: ", SwingUtilities.CENTER);
    LbEncontrar.setFont(new Font("Arial", Font.BOLD, 14));
    PanelCentro.add(LbEncontrar, BorderLayout.NORTH);

    JTextArea Area = new JTextArea();
    Area.setEditable(false);
    PanelCentro.add(Area, BorderLayout.CENTER);

    PanelPrincipal.add(panelBuscar, BorderLayout.CENTER);
    PanelPrincipal.add(btBuscar, BorderLayout.SOUTH);
    add(PanelPrincipal, BorderLayout.NORTH);
    add(PanelCentro, BorderLayout.CENTER);

    btBuscar.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            buscarPersonajes(); //ya te lo sabes no.. un metodo allí abajo
        }
    });
}

//un chapeo de rayo
String[] datos = new String[5];
if (vicDerr != null && vicDerr.contains("\n")) {
    datos = vicDerr.split("\n");
} else {
    datos[0] = "0";
    datos[1] = "0";
    datos[2] = "NA";
    datos[3] = "...";
    datos[4] = "...";
}

String victorias = datos.length > 0 ? datos[0].trim() : "0";
String derrotas = datos.length > 1 ? datos[1].trim() : "0";
String turnos = datos.length > 2 ? datos[2].trim() : "NA";
String fecha = datos.length > 3 ? datos[3].trim() : "...";
String hora = datos.length > 4 ? datos[4].trim() : "...";

resultado.append("Historial: ").append(victorias).append(" Derrotas").append(derrotas).append("\n");
resultado.append("Último Combate: ").append(turnos).append(" turnos. ").append(fecha).append(" Fecha: ").append(hora).append("\n");

Area.setText(resultado.toString());
validacion.registrarAcción("Buscar el personaje", true, "Salio bien-p");
} else {
    JOptionPane.showMessageDialog(this, "No hemos encontrado al personaje... no puiste noda verdad?", "Error 029", JOptionPane.ERROR_MESSAGE);
    validacion.registrarAcción("Buscar personajes", false, "Salio MAL");
}
}
}

```

Seguimos con la parte de gestión de archivos, qué corresponde a guardar, cargar y limpiar el estado del sistema, para esto se hicieron clases públicas, para su respectivo llamado en cada botón, primero tenemos de guardar personajes que se va a encargar de un archivo de texto con la información de los personajes ya registrados guardar los datos de los mismos junto con las victorias y rosas de ellos, tenemos la opción de cargar que nos va a permitir al cerrar el programa volver a recuperar los datos que ya habíamos registrado anteriormente buscando el archivo de texto, y por último tenemos la parte de limpiar el archivo del sistema, de igual manera buscando el archivo y sobrescribiendo en él para borrar todos los datos registrados.

```

public class gestorArchivo {

    private static final String NombreArchi = "Historial_personajes.txt";

    //ya me conseeeeeee, el jodido cafe ya no me hace efecto
    public static void GuardarPersonajes () {
        try (PrintWriter escribir = new PrintWriter(new FileWriter(NombreArchi))) {
            escribir.println(Personajes.CantPersonajes);
            for (int i = 0; i < Personajes.CantPersonajes; i++) {
                escribir.println(String.join(" ", Personajes.personaje[i]));
            }
            JOptionPane.showMessageDialog(null, "Se ha hecho guardado los Datos :D");
            validarAccion.regisAccion ("guardar datos del personaje", true, "datos de " + Personajes.CantPersonajes + "Salio bien");
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "No se pudo guardar, porque? yo que se man D" + e.getMessage(), "Error 030", JOptionPane.ERROR_MESSAGE);
            validarAccion.regisAccion ("guardar datos del personaje", false, "datos de " + Personajes.CantPersonajes + "Salio MAL");
        }
    }

    public static void CargarPersonajes () {
        //para buscar el archivo y buscar cada dato xd
        File archivo = new File(NombreArchi);
        if (archivo.exists()) {
            JOptionPane.showMessageDialog(null, "No hemos encontrado nada... Estamos Jodidos", "Error 031", JOptionPane.WARNING_MESSAGE);
            validarAccion.regisAccion ("Encontrar el archivo", false, "Salio MAL");

            return;
        }
        try (BufferedReader lector = new BufferedReader(new FileReader(archivo))) {
            String Lineas;
            if ((Lineas = lector.readLine()) != null) {
                Personajes.CantPersonajes = Integer.parseInt(Lineas.trim());
            }
            for (int i = 0; i < Personajes.CantPersonajes; i++) {
                if ((Lineas = lector.readLine()) != null) {
                    String[] datos = Lineas.split("\\\\");
                    if (datos.length >= Personajes.NUMCAMPSPERSONAJE) {
                        for (int j = 0; j < Personajes.NUMCAMPSPERSONAJE; j++) {
                            Personajes.personaje[i][j] = datos[j].trim();
                        }
                    }
                }
            }
        }
        JOptionPane.showMessageDialog(null, "Se ha Cargado el sistema :D");
        validarAccion.regisAccion ("guardar datos en archivo", true, "Salio MAL");
    } catch (IOException | NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "No se pudo cargar el archivo... SIP NOS JODIMOS" + e.getMessage(), "Error 032", JOptionPane.ERROR_MESSAGE);
        validarAccion.regisAccion ("guardar datos en archivo", false, "Salio MAL");
    }
}

public static void BorrarDatos () {
    File archivo = new File(NombreArchi);
    if (archivo.exists()) {
        int Confirm = JOptionPane.showConfirmDialog(null, "Se va a eliminar todos los datos...¿ REALMENTE ESTAS SEGURO?", "Confirmar Limpieza", JOptionPane.YES_NO_OPTION);
        if (Confirm == JOptionPane.YES_OPTION) {
            if (archivo.delete()) {
                Personajes.CantPersonajes = 0; //reseteamos el contador de personajes
                JOptionPane.showMessageDialog(null, "TODO fue Borrado... yay.");
                validarAccion.regisAccion ("borrar datos del archivo", true, "Salio bien");
            } else {
                JOptionPane.showMessageDialog(null, "Algo salio mal.. aun..o bueno ya sabes ", "Error 033", JOptionPane.ERROR_MESSAGE);
                validarAccion.regisAccion ("borrar datos del historial", false, "Salio MAL");
            }
        } else {
            JOptionPane.showMessageDialog(null, "No hay nada que borrar aun.. ", "Error 034", JOptionPane.WARNING_MESSAGE);
            validarAccion.regisAccion ("Borrar datos del historial", false, "Salio MAL");
        }
    }
}

```

Ya casi terminando tenemos la parte de la bitacora, el primer método de esto se encargara de registrar en cada acción salga bien o salga mal, en todas las clases creadas, esto se va a encargar de registrar la acción la hora y la fecha y si salió bien o salió mal esto imprimiéndose directamente en Consola inmediatamente cuando la acción se haya realizado y el segundo método que tenemos nos va a permitir ver las acciones en tiempo real tanto en consola como un archivo por aparte de texto esto gracias a filereader, printWriter y file.io, junto a sus debidas validaciones que nos va a permitir ver las acciones del usuario.

```

public class validarAcción {
    //una malausque herramienta que nos ayudara mas tarde

    private static final String NombreBit = "Bitacora_Acciones.txt"; //un archivo de datos
    private static final SimpleDateFormat FECHA = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    public static void regisAcción (String accion, boolean estado, String mensaje) {
        //ibtenemos la fecha
        String tiempo = FECHA.format(new Date());

        // ahora veremos male sal o sale bien
        String Estado = estado ? "Salio bien" : "Salio mal";

        //creamos un registro
        String registrar = String.format ("[%s] [%s] - %s: %s", tiempo, Estado, accion, mensaje);

        //lo podemos en consola
        System.out.println(registrar);

        //lo mandamos a un txt xddd por si las moscas
        try (PrintWriter escribir = new PrintWriter(new FileWriter(NombreBit, true))) {
            escribir.println(registrar);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Algo Malo sal... no se que chucha fue", "Error 030", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción ("guardar datos en la bitacora", false, "Salio mal");
        }
    }

    //ahora veremos la bitacora en consola
    public static void MostrarBit () {
        File archivo = new File(NombreBit);
        //ojala que no se vea feo despues del archivo de musica o si no me mato, se va a ver TODAS LAS LLAMADAS QUE HE HECHO DE ESTE METODO, no este pero ya sabes... enserio estas leyedo esto...
        System.out.println("\n" + " TODO LO QUE HA ESTADO PASANDO MIENTRAS SE EJECUTA ESTA COSA : " + "\n");

        if (!archivo.exists()){
            JOptionPane.showMessageDialog(null, "La bitacora no existe xd, ojala no la hayas borrado meno", "Error 031", JOptionPane.ERROR_MESSAGE);
            validarAcción.regisAcción ("Buscar bitacora", false, "Salio mal APROPOSITO");
            return;
        }
        //para que se borre eso
        int confirm = JOptionPane.showConfirmDialog(null, "¿se va a borrar la bitacora... estas seguro?", "Confirmar borrar de la botacora", JOptionPane.YES_NO_OPTION);

        if (confirm == JOptionPane.YES_OPTION){
            if (archivo.delete()){
                JOptionPane.showMessageDialog(null, "Todas las acciones se borraron :D", "salio bien", JOptionPane.INFORMATION_MESSAGE);
                validarAcción.regisAcción ("Borrar la bitacora", true, "Salio bien");
            } else {
                JOptionPane.showMessageDialog(null, "No borramos la bitacora, porque? no me pregantes wn tu lo decidiste", "Error 032", JOptionPane.ERROR_MESSAGE);
                validarAcción.regisAcción ("borrar la bitacora", true, "Salio y ya");
            }
        }
    }
}

```

Como último tenemos la clase más importante que es la demostrar mis datos de estudiante que estos son es una ventana creada con JPanel, agregando al panel principal el texto de mi nombre mi carnet y un mensaje divertido, el título está puesto con una tipografía diferente al resto de ventanas junto con la pantalla de bienvenida.

```
public class YOOOOO extends JFrame {  
  
    public YOOOOO(){  
        //ya no quiero explicar esto  
        setTitle("SOY YOOOOOOOO");  
        setSize(500, 300);  
        setLocationRelativeTo(null);  
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
  
        JPanel panelPrincipal = new JPanel(new GridLayout(6, 1, 10, 10));  
        //el título  
        JLabel titulo = new JLabel("MIS DATOS :3", SwingConstants.CENTER);  
        titulo.setFont(new Font("Papyrus", Font.BOLD, 25));  
        panelPrincipal.add(titulo);  
  
        panelPrincipal.add(new JLabel("Mi nombre es: Guillermo Enrique Marroquin Morán", SwingConstants.CENTER));  
        panelPrincipal.add(new JLabel("Mi Carnet es 202103527", SwingConstants.CENTER));  
        panelPrincipal.add(new JLabel("Este programa tiene kilos y kilos de copy :3, mas la musica de undertale xdxdxd"));  
  
        //botón para que se cierre esta madre  
        JButton btcerrar = new JButton("salir");  
        btcerrar.addActionListener(e -> dispose());  
        panelPrincipal.add(btcerrar);  
  
        add(panelPrincipal);  
        validarAccion.regisAccion("Ver mis datos", true, "Salio bien, como demonios debe de salir mal esto?");  
    }  
}
```

¿Qué problemas tuve con la práctica?

Al realizar esta práctica de programación tuve más problemas del que debería tener 😞.

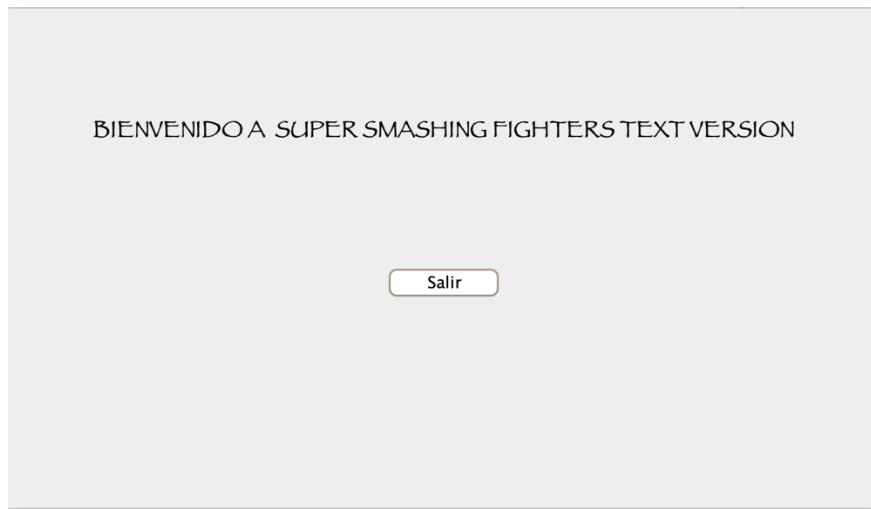
- A la hora de agregar una acción a cada botón el programa DESPUES de realizar la acción, salía en pantalla “Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException”
- A la hora de querer modificar el ataque del personaje, aparecía el mensaje de qué el ataque de este debe de estar en el rango de 10 a 100, incluso al estar así.
- A la hora de querer buscar un personaje, la acción de buscar el personaje de dedicada a su botón no funcionaba.
- A la hora de querer mostrar en el historial de combates, la hora y fecha no se mostraba cómo se debería.
- A la hora de querer hacer un contador de turnos el programa se caía.
- Cuando terminaba un combate el programa se caía.
- Cuando agregaba una nueva ventana o elementos a una ventana nada se veía en pantalla.

¿Cómo se pudo solucionar los problemas

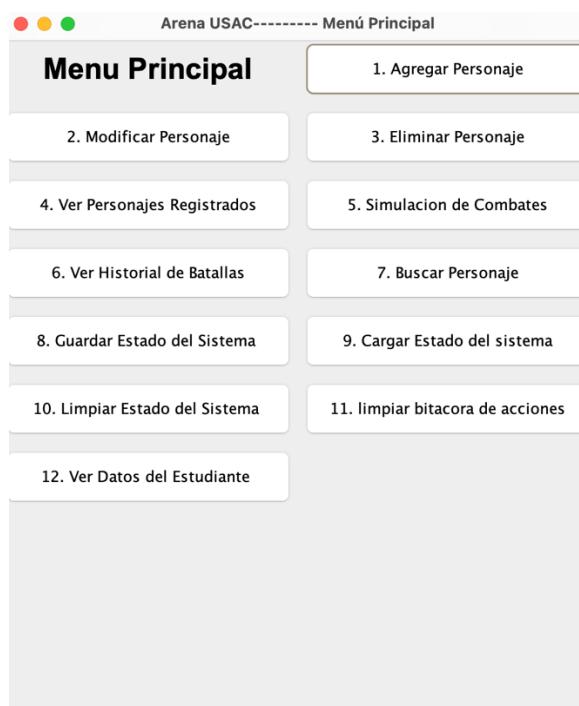
- En cada botón después de la acción y saliera ese mensaje en pantalla, debíamos dirigirnos a la línea donde estaba el problema, y empezar a corregir el código otra vez.
- En este problema fue mas un descuido, qué enlazaba la condición de rango de los puntos de vida con los puntos de ataque, por lo que sólo teníamos que separar ambas condiciones.
- Este problema fue más de otro descuido dado que la acción estaba más el botón no estaba configurado (por no decir que se me olvido poner la jodida acción y el botón en sí).
- A la hora de querer contabilizar los turnos de un combate lo que se hizo fue hacer un contador, en el bucle while, y en la condición if dónde por cada acción y cada esquivada cuente como un turno.
- Cuando se terminaba el combate el programa sé que haya dado que registraba el resultado de esta y el separador entre las condiciones de la victoria estaban por guiones lo que hice fue usar el trim y Split para separar en pantalla las “condiciones del combate y victoria”.
- A la hora de cuando se agrega elementos nuevos a una ventana existente muchas veces estos se tapan entre sí, por lo que me tocó que usar el: FlowLayout, para repartir los elementos en una ventana sin que esto se tapen unos a otros.

Menú interactivo

Pantalla de bienvenida:



Menú principal:



Agregar personaje:

The screenshot shows the 'Agregar Personaje' (Add Character) window. It contains fields for character details:

- Escribe el nombre que quieras: Mario
- Elige el arma de tu preferencia: Guantes de algodón
- Escribe los puntos de vida del personaje (100 a 500): 300
- Escribe los puntos de ataque del personaje (10 a 100): 45
- Inserta la velocidad del personaje (1 a 10): 5
- Escriba la agilidad del personaje (1 a 10): 4
- Escribe la defensa que tendrá el personaje (1 a 50): 25

At the bottom left is a 'Guardar cambios' (Save changes) button. A message window titled 'Mensaje' (Message) appears, stating 'Mario se ha agregado al roster de personajes' (Mario has been added to the character roster) with an 'Aceptar' (Accept) button.

Modificar personaje:

The screenshot shows the 'Modificar Personaje' (Modify Character) window. It displays current character stats and allows modification of specific fields:

Nombre:	Arma:	Puntos de vida:
Mario	Guantes de algodón	300
Ataque:	Velocidad:	Agilidad:
45	5	4
Defensa:		
25		
Escoge el arma de tu preferencia:	Ingresá los puntos de vida (100 a 500):	
Guantes de algodón	300	25
Ingresá los puntos de ataque (10 a 100):	Ingresá los puntos de velocidad (1 a 10):	Ingresá los puntos de agilidad (1 a 10):
45	5	4
Ingresá los puntos de defensa:		
25		

At the bottom left is a 'Guardar Cambios' (Save changes) button.

Ilustración 1: Antes de modificar personaje

Modificar Personaje

Modificar personaje

Buscar por ID o Nombre

1 Buscar

Nombre: Mario Arma: Guantes de algodón Puntos de vida: 300

Ataque: 45 Velocidad: 5 Agilidad: 4

Defensa: 25

Escoge el arma de tu preferencia: Guantes de algodón Ingresá los puntos de vida (100 a 500):

250 Ingresá los puntos de ataque (10 a 100) 30

Ingresá los puntos de velocidad (1 a 10) 5 Ingresá los puntos de agilidad (1 a 10)

3 Ingresá los puntos de defensa: 30

Ilustración 2: Despues de modificar al personaje

Eliminar personaje:

Eliminar Personaje

Eliminar Personaje Escribe ACA el ID o el nombre del personaje que deseas eliminar

MARIO Buscar

El personaje a eliminar es: Mario Eliminar Personaje

Visualizar personaje:

Personajes Registrados								
ID	Nombre	Arma	Hp	Ataque	Velocidad	Agilidad	Defensa	Historial
1	Mario	Guantes de ...	250	30	5	3	30	0-0
2	Luigi	Guantes de ...	300	25	6	4	20	0-0

Simulación de combates:



Ilustración 3: Antes del combate

Simulador de Combates

Elige dos personajes para Luchar Elige al primer luchador Mario Ahora Elige al segundo luchador Luigi MADRAZOS

```

10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 150
Turno14: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 140
Turno15: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno15: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 130
Turno16: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 120
Turno16: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 110
Turno17: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno17: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 100
Turno18: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 90
Turno18: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 80
Turno19: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 70
Turno19: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 60
Turno20: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 50
Turno20: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno21: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 40
Turno21: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno22: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno22: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno23: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 30
Turno23: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno24: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 20
Turno24: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 10
Turno25: Mario HA ATACADO, PERO
LuigiHA ESQUIVADO EL ATAQUE
Turno25: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO
10 PUNTOS DE DAÑO, LOS DE VIDA RESTANTES DE Luigi ES: 0
Mario Ha ganado el combate en
25turnos

```

Mario Ha ganado el combate en
25turnos

Ilustración 4: Despues del combate

Ver historial de combates:

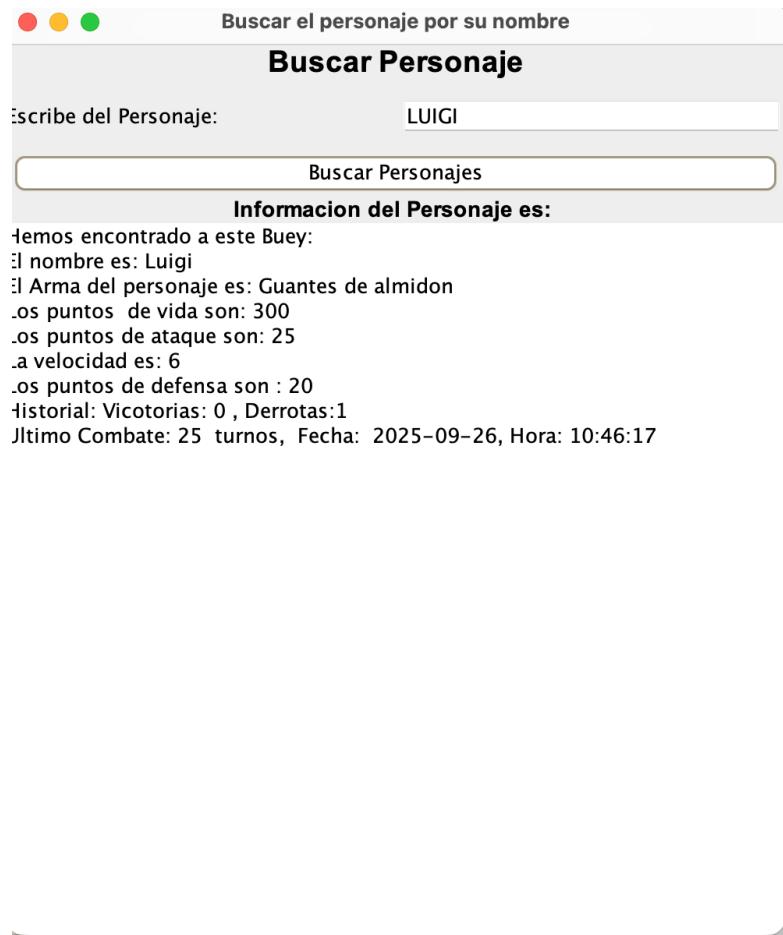
Este es el historial de combates realizados

Historial de Victoria y Derrotas Última actualización: 2025-09-26 10:47:00

Personaje	Victorias	Pérdidas	Último Combate	Término	Hora
Mario	0	0	Turno25: Mario HA ATACADO, PERO	LuigiHA ESQUIVADO EL ATAQUE	10:46:17
Luigi	1	1	Turno25: Mario HA ECHO UN ATAQUE Luigi HA SURRIDO	Mario Ha ganado el combate en	10:46:17

Actualizar historial

Buscar Personaje:



Guardar el estado de combate:



```
Historial_personajes.txt ~
2
1|Mario|Guantes de algodon|250|30|5|3|30|6 | 0 | 25|2025-09-26 | 10:46:17
2|Luigi|Guantes de almidon|300|25|6|4|20|0 | 1 | 25|2025-09-26 | 10:46:17
```

Cargar el estado del sistema:



Ver bitácora de acciones:

```
Reproduciendo:Song_That_Plays_When_Somebody_Verse_Sans_(by_Carlos_Insaneintherain_Eiene)
De Deltarune, por Toby Fox y compañía
[2025-09-26 10:23:13] [Salio bien] - Escuchar un temazo: Salio bien
La musica se ha detenido...
[2025-09-26 10:27:02] [Salio bien] - agregar personaje: Salio bien
[2025-09-26 10:32:42] [Salio bien] - encontrar el personaje para eliminarlo: Salio bien
[2025-09-26 10:33:39] [Salio bien] - Eliminar el personaje: Salio... y ya
[2025-09-26 10:36:05] [Salio bien] - agregar personaje: Salio bien
[2025-09-26 10:41:48] [Salio bien] - madrazos entre personajes: Salio bien
[2025-09-26 10:41:48] [Salio bien] - madrazos entre personajes: Salio bien
[2025-09-26 10:46:10] [Salio bien] - madrazos entre personajes: Salio bien
[2025-09-26 10:46:10] [Salio bien] - madrazos entre personajes: Salio bien
[2025-09-26 10:46:17] [Salio bien] - madrazos entre personajes: Salio bien
[2025-09-26 10:46:17] [Salio bien] - madrazos entre personajes: Salio bien
[2025-09-26 10:46:45] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:45] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:55] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:55] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:57] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:57] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:57] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:57] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:57] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:58] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:58] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:58] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:58] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:58] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:58] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:59] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:46:59] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:47:00] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:47:00] [Salio bien] - ver historial de madrazos entre personajes: Salio bien
[2025-09-26 10:48:09] [Salio bien] - Buscar el personaje: Salio bien ;p
[2025-09-26 10:52:00] [Salio bien] - guardar datos del personaje: datos de 2Salio bien
[2025-09-26 10:52:04] [Salio bien] - guardar datos del personaje: datos de 2Salio bien
[2025-09-26 10:56:04] [Salio bien] - guardar datos en archivo: Salio MAL
[2025-09-26 10:57:51] [Salio bien] - guardar datos del personaje: datos de 2Salio bien
[2025-09-26 10:57:54] [Salio bien] - guardar datos en archivo: Salio MAL
[2025-09-26 10:57:58] [Salio bien] - guardar datos en archivo: Salio MAL
```

Datos del estudiante:



EGRAFIA

- Escobar. (2021, September 16). *Cómo crear una ventana y ejecutar el proyecto* [Video]. YouTube. Retrieved September 26, 2025, from <https://www.youtube.com/watch?v=dyhELSYQpII>
- Lopez, A. (2019, February 27). *Como reproducir un sonido*. Stack Overflow En Español. Retrieved September 26, 2025, from <https://es.stackoverflow.com/questions/241578/como-reproducir-un-sonido>
- Padrón, J. M. G. (2018, season-02). *Como Crear Ventana Java manualmente facilmente y sin layouts con Control de Eventos y ActionListener*. Retrieved September 26, 2025, from <https://jmguijera.blogspot.com/2017/03/crear-una-ventana-manualmente-con-jav.html>
- applikdos. (2014, February 26). *Escuchar Eventos | Acciones sobre Botones en Java | Tutorial de Java* [Video]. YouTube. Retrieved September 26, 2025, from <https://www.youtube.com/watch?v=zj0Whq2laIE>
- ProgramaTutos. (2023, October 3). *Como Reproducir Sonidos (wav) en Java* [Video]. YouTube. Retrieved September 26, 2025, from <https://www.youtube.com/watch?v=OvVBlrLiNf4>
- RC, J. (2016, June 5). *Botón funcional en Java*. Stack Overflow En Español. Retrieved September 26, 2025, from <https://es.stackoverflow.com/questions/12907/bot%C3%B3n-funcional-en-java>
- De La Torre, F. a. C. (2017, season-03). *Introduccion a graficos mediante JAVA 2D*. Slideshare. Retrieved September 26, 2025, from <https://es.slideshare.net/slideshow/introduccion-a-graficos-mediante-java-2d/76195721>
- Facultado de ingeniería. (2016, March 1). *Creacionistas de interfaces graficas*. Udb. Retrieved September 26, 2025, from https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/java-avanzado/2019/i/quia-5.pdf
- Academia Hormiga. (2020, November 27). *Como hacer un boton que abra una ventana en Java* [Video]. YouTube. Retrieved September 26, 2025, from <https://www.youtube.com/watch?v=h2oTM6ehU-E>
- RRGT19. (2016, December 3). *Cómo puedo cerrar una ventana en Java y que aparezca la ventana anterior que la llamó?* Stack Overflow En Español. Retrieved September 26, 2025, from <https://es.stackoverflow.com/questions/37403/c%C3%B3mo-puedo-cerrar-una-ventana-en-java-y-que-aparezca-la-ventana-anterior-que-la>

- Programando con Jhovany. (2023, January 3). *Cambiar de Ventana en Java* [Video]. YouTube. Retrieved September 26, 2025, from <https://www.youtube.com/watch?v=N0JhuCZFaJ4>
- GeeksforGeeks. (2021, July 22). *Java Swing | JSeparator with examples*. GeeksforGeeks. Retrieved September 26, 2025, from https://www.geeksforgeeks.org.translate.goog/java/java-swing-jseparator-with-examples/?x_tr_sl=en&x_tr_tl=es&x_tr_hl=es&x_tr_pto=tc
- Editor pagina. (2017, season-03). *Swing en Java*. Bahiaxit. Retrieved September 26, 2025, from <https://bahiaxit.com/entrada/swing-en-java>
- Roberto HArellano. (2020, November 9). *Utiliza Java y Swing para trabajar con JTable y JScrollPane* [Video]. YouTube. Retrieved September 26, 2025, from https://www.youtube.com/watch?v=L_UfthdgKgRU
- DataCamp team. (2019, season-02). *Hilos de Java*. DataCamp. Retrieved September 26, 2025, from <https://www.datacamp.com/es/doc/java/threads>
- River. (2011, November 18). *Probability in Java*. Stack Overflow. Retrieved September 26, 2025, from <https://stackoverflow.com/questions/8183840/probability-in-java>
- Obregon, A. (2024, August 25). *Java's Thread.sleep() Method Explained*. Medium. Retrieved September 26, 2025, from <https://medium.com/@AlexanderObregon/javas-thread-sleep-method-explained-8e1f4df3e548>
- <https://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?codigo=109&punto=&inicio=20>
- SolarWinds Loggly. (2025, June 27). *Java Logging Basics - the ultimate guide to logging*. Log Analysis | Log Monitoring by Loggly. Retrieved September 26, 2025, from https://www-loggly-com.translate.goog/ultimate-guide/java-logging-basics/?x_tr_sl=en&x_tr_tl=es&x_tr_hl=es&x_tr_pto=sqe
- Nivardo. (2024, September 12). *JScrollPane en Java*. Oregoom.com. Retrieved September 26, 2025, from <https://oregoom.com/java/jscrollpane/>