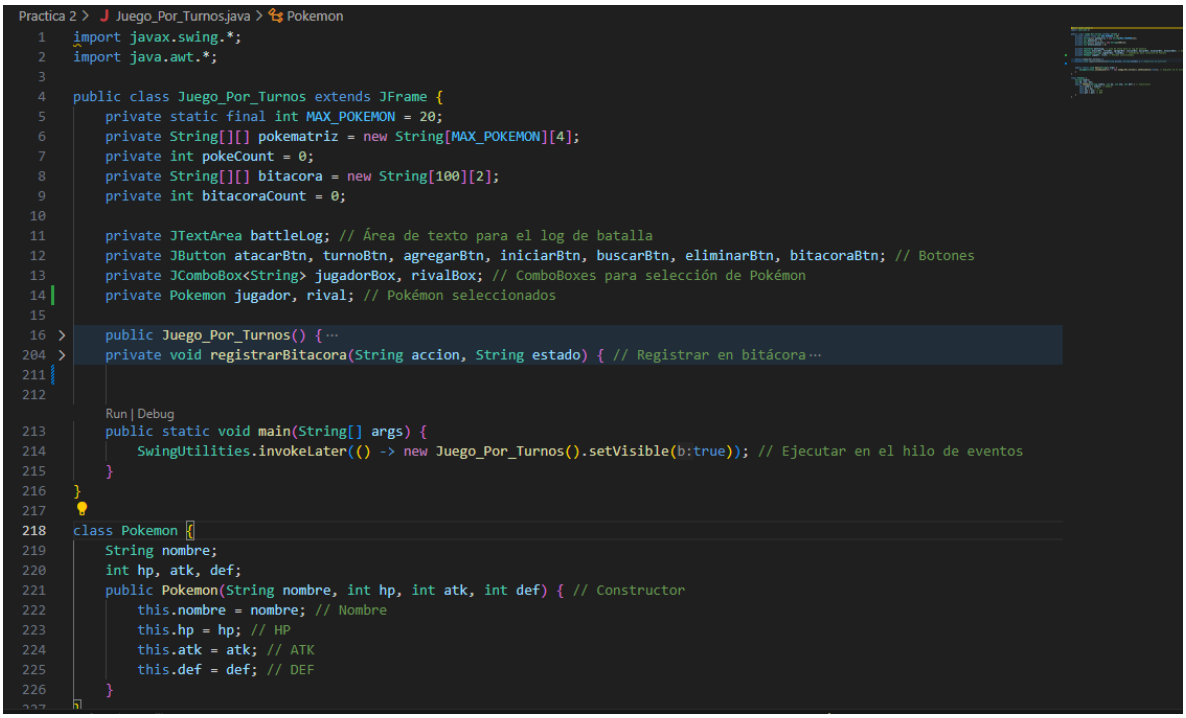
Manual técnico

Caratula de Codigo (Abreviado)



```
Practica 2 > Juego_Por_Turnos.java > Pokemon
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class Juego_Por_Turnos extends JFrame {
5       private static final int MAX_POKEMON = 20;
6       private String[][] pokematriz = new String[MAX_POKEMON][4];
7       private int pokeCount = 0;
8       private String[][] bitacora = new String[100][2];
9       private int bitacoraCount = 0;
10
11      private JTextArea battleLog; // Área de texto para el log de batalla
12      private JButton atacarBtn, turnoBtn, agregarBtn, iniciarBtn, buscarBtn, eliminarBtn, bitacoraBtn; // Botones
13      private JComboBox<String> jugadorBox, rivalBox; // ComboBoxes para selección de Pokémon
14      private Pokemon jugador, rival; // Pokémon seleccionados
15
16      public Juego_Por_Turnos() {···
204     private void registrarBitacora(String accion, String estado) { // Registrar en bitácora···
211
212
        Run | Debug
213     public static void main(String[] args) {
214         SwingUtilities.invokeLater(() -> new Juego_Por_Turnos().setVisible(b:true)); // Ejecutar en el hilo de eventos
215     }
216 }
217
218 class Pokemon {
219     String nombre;
220     int hp, atk, def;
221     public Pokemon(String nombre, int hp, int atk, int def) { // Constructor
222         this.nombre = nombre; // Nombre
223         this.hp = hp; // HP
224         this.atk = atk; // ATK
225         this.def = def; // DEF
226     }
```

Diccionario de Metodos

Librerias

java.swing.*: Permite crear y manejar componentes de interfaz gráfica como ventanas, botones, etiquetas, campos de texto, áreas de texto, ComboBoxes y cuadros de diálogo.

java.awt.*: Proporciona clases para el diseño y organización visual de los componentes gráficos, como layouts y colores.

Metodos de Codigos

public Juego_Por_Turnos(): Constructor de la clase principal. Inicializa la ventana, los paneles, los campos de texto, botones, ComboBoxes y el área de log. Configura todos los listeners para los botones y define la lógica de interacción de la interfaz gráfica.

private void registrarBitacora(String accion, String estado): Registra una acción y su estado (correcto/error) en la matriz de bitácora. Se utiliza para llevar un historial de las operaciones realizadas en el sistema (agregar, buscar, eliminar, iniciar batalla, etc.).

public static void main(String[] args): Método principal de ejecución. Inicia la aplicación gráfica creando una instancia de la ventana principal y ejecutándola en el hilo de eventos de Swing.

class Pokemon { }: Clase que representa un Pokémon.Contiene los atributos nombre, hp, atk y def, y su constructor para inicializar estos valores.

agregarBtn.addActionListener(e -> { }): Acción para agregar Pokémon.

buscarBtn.addActionListener(e -> { }): Acción para buscar Pokémon.

eliminarBtn.addActionListener(e -> { }): Acción para eliminar Pokémon.

bitacoraBtn.addActionListener(e -> { }): Acción para mostrar la bitácora.

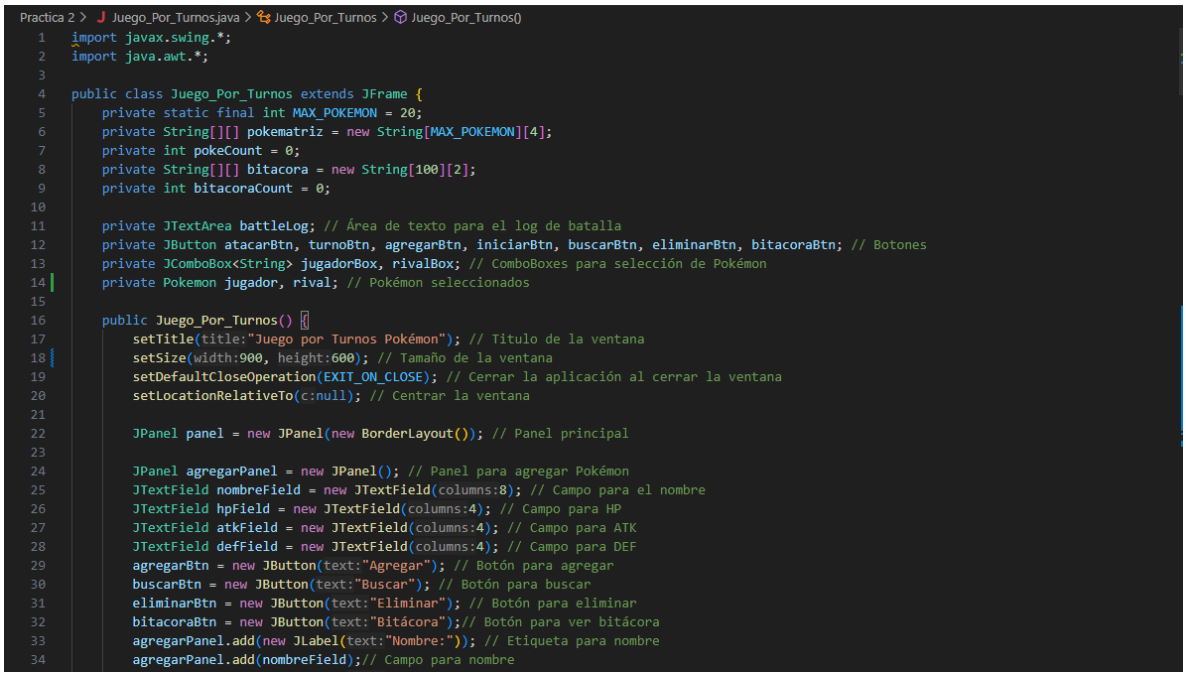
iniciarBtn.addActionListener(e -> { }): Acción para iniciar la batalla.

atacarBtn.addActionListener(e -> { }): Acción para el turno de ataque del jugador.

turnoBtn.addActionListener(e -> { }): Acción para el turno de ataque del rival.

private void registrarBitacora(String accion, String estado): Registra una acción y su estado en la matriz de bitácora.

Codigo completo



```
JPanel agregarPanel - Juego_Por_Turnos.Juego_Por_Turnos()
        agregarPanel.add(hpField);
        agregarPanel.add(new JLabel(text:"ATK:"));
        agregarPanel.add(atkField);
        agregarPanel.add(new JLabel(text:"DEF:"));
        agregarPanel.add(defField);
        agregarPanel.add(agregarBtn); // Botón para agregar
        agregarPanel.add(buscarBtn); // Botón para buscar
        agregarPanel.add(eliminarBtn); // Botón para eliminar
        agregarPanel.add(bitacoraBtn); // Botón para ver bitácora

        JPanel seleccionPanel = new JPanel(); // Panel para selección de Pokémon
        jugadorBox = new JComboBox<>(); // ComboBox para el jugador
        rivalBox = new JComboBox<>(); // ComboBox para el rival
        iniciarBtn = new JButton(text:"Iniciar Batalla"); // Botón para iniciar batalla
        seleccionPanel.add(new JLabel(text:"Tu Pokémon:")); // Etiqueta para el jugador
        seleccionPanel.add(jugadorBox); // ComboBox para el jugador
        seleccionPanel.add(new JLabel(text:"Rival:")); // Etiqueta para el rival
        seleccionPanel.add(rivalBox); // ComboBox para el rival
        seleccionPanel.add(iniciarBtn); // Botón para iniciar batalla

        battleLog = new JTextArea(rows:10, columns:40); // Área de texto para el log de batalla
        battleLog.setEditable(b:false); // No editable
        JScrollPane scroll = new JScrollPane(battleLog); // Scroll para el área de texto

        JPanel batallaPanel = new JPanel(); // Panel para botones de batalla
        atacarBtn = new JButton(text:"Atacar"); // Botón para atacar
        turnoBtn = new JButton(text:"Turno Rival"); // Botón para turno del rival
        atacarBtn.setEnabled(b:false);  // Inicialmente deshabilitado
        turnoBtn.setEnabled(b:false); // Inicialmente deshabilitado
        batallaPanel.add(atacarBtn); // Botón para atacar
        batallaPanel.add(turnoBtn); // Botón para turno del rival
        panel.add(agregarPanel, BorderLayout.NORTH); // Agregar panel en la parte superior
        panel.add(seleccionPanel, BorderLayout.CENTER); // Agregar panel en el centro
        panel.add(batallaPanel, BorderLayout.EAST); // Agregar panel a la derecha
        panel.add(scroll, BorderLayout.SOUTH); // Agregar área de texto en la parte inferior

        add(panel); // Agregar panel principal a la ventana

        // La "e" es para que se resica informacion del evento (click del boton)
        agregarBtn.addActionListener(e -> { // Acción para agregar Pokémon
            try {
                String nombre = nombreField.getText().trim(); // Obtener nombre
                int hp = Integer.parseInt(hpField.getText()); // Convertir HP a entero
                int atk = Integer.parseInt(atkField.getText()); // Convertir ATK a entero
                int def = Integer.parseInt(defField.getText()); // Convertir DEF a entero
                if (nombre.isEmpty() || hp <= 0 || atk < 0 || def < 0 || pokeCount >= MAX_POKEMON) throw new Exception();
                pokematriz[pokeCount][0] = nombre; // Guardar nombre
                pokematriz[pokeCount][1] = String.valueOf(hp); // Guardar HP como cadena
                pokematriz[pokeCount][2] = String.valueOf(atk); // Guardar ATK como cadena
                pokematriz[pokeCount][3] = String.valueOf(def); // Guardar DEF como cadena
                jugadorBox.addItem(nombre); // Agregar al ComboBox del jugador
                rivalBox.addItem(nombre); // Agregar al ComboBox del rival
                pokeCount++;
                battleLog.append("Pokémon agregado: " + nombre + "\n");
                nombreField.setText(t:""); hpField.setText(t:""); atkField.setText(t:""); defField.setText(t:"");
                registrarBitacora(accion:"Agregar Pokémon", estado:"Correcto");
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this, message:"Datos inválidos o límite alcanzado.");
                registrarBitacora(accion:"Agregar Pokémon", estado:"Error");
            }
        });

        buscarBtn.addActionListener(e -> { // Acción para buscar Pokémon
            String nombre = JOptionPane.showInputDialog(this, message:"Nombre a buscar:");
            boolean encontrado = false; // Bandera para encontrar
            for (int i = 0; i < pokeCount; i++) { // Recorrer matriz
                if (pokematriz[i][0].equalsIgnoreCase(nombre)) { // Comparar nombres
                    battleLog.append("Encontrado: " + pokematriz[i][0] +
                        " HP:" + pokematriz[i][1] +
                        " ATK:" + pokematriz[i][2] +
                        " DEF:" + pokematriz[i][3] + "\n");
                    encontrado = true;
                }
            }
            if (!encontrado) battleLog.append(str:"No se encontró el Pokémon.\n");
            registrarBitacora(accion:"Buscar Pokémon", encontrado ? "Correcto" : "Error");
        });
```

```java
        eliminarBtn.addActionListener(e -> { // Acción para eliminar Pokémon
            String nombre = JOptionPane.showInputDialog(this, message:"Nombre a eliminar:");
            boolean eliminado = false; // Bandera para eliminar
            for (int i = 0; i < pokeCount; i++) { // Recorrer matriz
                if (pokematriz[i][0].equalsIgnoreCase(nombre)) { // Comparar nombres
                    for (int j = i; j < pokeCount - 1; j++) pokematriz[j] = pokematriz[j + 1]; // Desplazar filas
                    pokematriz[pokeCount - 1] = new String[4]; // Limpiar última fila
                    jugadorBox.removeItemAt(i); // Eliminar del ComboBox del jugador
                    rivalBox.removeItemAt(i); // Eliminar del ComboBox del rival
                    pokeCount--;
                    battleLog.append("Eliminado: " + nombre + "\n");
                    eliminado = true;
                    break;
                }
            }
            if (!eliminado) battleLog.append(str:"No se encontró el Pokémon para eliminar.\n");
            registrarBitacora(accion:"Eliminar Pokémon", eliminado ? "Correcto" : "Error");
        });

        bitacoraBtn.addActionListener(e -> { // Acción para ver bitácora
            StringBuilder sb = new StringBuilder(str:"--- Bitácora ---\n");
            for (int i = 0; i < bitacoraCount; i++) // Recorrer bitácora
                sb.append(bitacora[i][0]).append(str:" - ").append(bitacora[i][1]).append(str:"\n"); // Agregar a StringBuilder
            JOptionPane.showMessageDialog(this, sb.toString()); // Mostrar bitácora
        });

        iniciarBtn.addActionListener(e -> {
            int idxJugador = jugadorBox.getSelectedIndex();
            int idxRival = rivalBox.getSelectedIndex();
            if (idxJugador == -1 || idxRival == -1 || idxJugador == idxRival) {
                JOptionPane.showMessageDialog(this, message:"Selecciona Pokémon válidos.");
                return;
            }
            jugador = new Pokemon(
                pokematriz[idxJugador][0],
                Integer.parseInt(pokematriz[idxJugador][1]),
                Integer.parseInt(pokematriz[idxJugador][2]),
                Integer.parseInt(pokematriz[idxJugador][3])
            );
            rival = new Pokemon(
                pokematriz[idxRival][0],
                Integer.parseInt(pokematriz[idxRival][1]),
                Integer.parseInt(pokematriz[idxRival][2]),
                Integer.parseInt(pokematriz[idxRival][3])
            );
            battleLog.append(str:"¡Comienza la batalla!\n");
            battleLog.append("Tu Pokémon: " + jugador.nombre + " (HP: " + jugador.hp + ")\n");
            battleLog.append("Rival: " + rival.nombre + " (HP: " + rival.hp + ")\n");
            atacarBtn.setEnabled(b:true);
            turnoBtn.setEnabled(b:false);
            registrarBitacora(accion:"Iniciar Batalla", estado:"Correcto");
        });

        atacarBtn.addActionListener(e -> {
            if (jugador.hp > 0 && rival.hp > 0) {
                int dano = Math.max(a:0, jugador.atk - rival.def);
                rival.hp -= dano;
                battleLog.append(jugador.nombre + " ataca y hace " + dano + " de daño a " + rival.nombre + "\n");
                battleLog.append("HP de " + rival.nombre + ": " + Math.max(a:0, rival.hp) + "\n");
                if (rival.hp <= 0) {
                    battleLog.append(str:"¡Ganaste!\n");
                    atacarBtn.setEnabled(b:false);
                    turnoBtn.setEnabled(b:false);
                    registrarBitacora(accion:"Fin Batalla", estado:"Ganaste");
                } else {
                    atacarBtn.setEnabled(b:false);
                    turnoBtn.setEnabled(b:true);
                }
            }
        });

        turnoBtn.addActionListener(e -> {
            if (rival.hp > 0 && jugador.hp > 0) {
                int dano = Math.max(a:0, rival.atk - jugador.def);
                jugador.hp -= dano;
                battleLog.append(rival.nombre + " ataca y hace " + dano + " de daño a " + jugador.nombre + "\n");
                battleLog.append("HP de " + jugador.nombre + ": " + Math.max(a:0, jugador.hp) + "\n");
                if (jugador.hp <= 0) {
```

```
                    battleLog.append(str:"¡Perdiste!\n");
                    atacarBtn.setEnabled(b:false);
                    turnoBtn.setEnabled(b:false);
                    registrarBitacora(accion:"Fin Batalla", estado:"Perdiste");
                } else {
                    atacarBtn.setEnabled(b:true);
                    turnoBtn.setEnabled(b:false);
                }
            }
        });
    }
    private void registrarBitacora(String accion, String estado) { // Registrar en bitácora
            if (bitacoraCount < 100) { // Limitar a 100 entradas
            bitacora[bitacoraCount][0] = accion; // Guardar acción
            bitacora[bitacoraCount][1] = estado; // Guardar estado
            bitacoraCount++;
            }
        }


    Run | Debug
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new Juego_Por_Turnos().setVisible(b:true)); // Ejecutar en el hilo de eventos
    }
}

class Pokemon {
    String nombre;
    int hp, atk, def;
    public Pokemon(String nombre, int hp, int atk, int def) { // Constructor
        this.nombre = nombre; // Nombre
        this.hp = hp; // HP
        this.atk = atk; // ATK
        this.def = def; // DEF
    }
}

```