

# Documentación de la Interfaz de Usuario - Juego Mancala

Juan Andrés Rodríguez Rubio  
Carlos Enrique Caicedo Guerrero  
Alejandro Caicedo Caicedo

## 1. Descripción General

Este documento presenta una descripción de la interfaz de usuario del juego clásico **Mancala**, implementado en Python para ser ejecutado desde la consola. El juego permite la participación de dos jugadores que se turnan para distribuir piedras y aplicar las reglas tradicionales del Mancala.

## 2. Reglas del Juego

- Cada jugador dispone de 6 pozos, cada uno con 4 piedras al inicio de la partida.
- Cada jugador posee una **Mancala** (depósito) donde acumula las piedras capturadas.
- Durante su turno, el jugador selecciona uno de sus pozos y distribuye sus piedras en sentido horario.
- Si la última piedra cae en su propia Mancala, el jugador obtiene un turno adicional.
- Si la última piedra cae en un pozo vacío del propio lado y el pozo opuesto contiene piedras, se realiza una **captura**.
- La partida finaliza cuando un jugador ya no tiene piedras en sus pozos.
- Gana el jugador que acumule la mayor cantidad de piedras en su Mancala.

## 3. Descripción del código

### 3.1. Diseño de la clase Mancala

Se diseñó una clase que está conformada por dos atributos:

- **board**: Lista que representa el tablero con 14 posiciones:
  - Posiciones 0-5: Pozos del jugador 1

- Posición 6: Mancala del jugador 1
  - Posiciones 7-12: Pozos del jugador 2
  - Posición 13: Mancala del jugador 2
- `player_turn`: Indica qué jugador tiene el turno (1 o 2)

### 3.1.1. Constructor `__init__`

Inicializa el tablero de juego y establece el turno del jugador 1.

```
def __init__(self):
    # Inicializar el tablero: 6 pozos para cada jugador y 1 mancala
    self.board = [4] * 6 + [0] + [4] * 6 + [0]
    self.player_turn = 1 # Jugador 1 comienza
```

### 3.1.2. Método `print_board`

Imprime por consola el estado actual del tablero.

```
def print_board(self):
    print("\nTablero:")
    print("Jugador 2 (Mancala: {})".format(self.board[13]))
    print("Posicion fichas: 12  11  10  9  8  7")
    print("Pozo jugador 2: ", self.board[12:6:-1])
    print("Pozo jugador 1 ", self.board[0:6])
    print("Posicion fichas: 0  1  2  3  4  5")
    print("Jugador 1 (Mancala: {})".format(self.board[6]))
    print("Turno del Jugador", self.player_turn)
```

### 3.1.3. Método `make_move`

Realiza un movimiento en el juego según el pozo seleccionado. En primer lugar, se valida si el movimiento está de acuerdo con las reglas establecidas. En caso contrario, se muestra un mensaje de error al usuario. Si el movimiento es válido, se colocan las piedras en los lugares adecuados y se hacen las verificaciones necesarias de acuerdo con las reglas del juego. Después de revisar si se aplican saltos de mancala o capturas de piedras, se pasa el turno al siguiente jugador.

```
def make_move(self, pit):
    # Validación del movimiento
    if self.player_turn == 1:
        if pit < 0 or pit > 5 or self.board[pit] == 0:
            print("Movimiento inválido. Intenta de nuevo.")
            return False
    else:
        if pit < 7 or pit > 12 or self.board[pit] == 0:
            print("Movimiento inválido. Intenta de nuevo.")
```

```

        return False

    # Distribución de las piedras
    stones = self.board[pit]
    self.board[pit] = 0
    current_pit = pit

    while stones > 0:
        current_pit = (current_pit + 1) % 14
        if (self.player_turn == 1 and current_pit == 13) or
            (self.player_turn == 2 and current_pit == 6):
            continue # Saltar el mancala del oponente
        self.board[current_pit] += 1
        stones -= 1

    # Captura de piedras
    if self.board[current_pit] == 1 and
        ((self.player_turn == 1 and 0 <= current_pit <= 5) or
         (self.player_turn == 2 and 7 <= current_pit <= 12)):
        opposite_pit = 12 - current_pit
        if self.board[opposite_pit] > 0:
            self.board[6 if self.player_turn == 1 else 13] +=
                self.board[opposite_pit] + 1
            self.board[current_pit] = 0
            self.board[opposite_pit] = 0

    # Cambio de turno
    if (self.player_turn == 1 and current_pit != 6) or
        (self.player_turn == 2 and current_pit != 13):
        self.player_turn = 3 - self.player_turn

    return True

```

#### 3.1.4. Método check\_game\_over

Verifica si el juego ha terminado (cuando todos los pozos de un jugador están vacíos).

```

def check_game_over(self):
    if all(stones == 0 for stones in self.board[0:6]) or
        all(stones == 0 for stones in self.board[7:13]):
        self.board[6] += sum(self.board[0:6])
        self.board[13] += sum(self.board[7:13])
        for i in range(14):
            if i != 6 and i != 13:
                self.board[i] = 0
        return True
    return False

```

### 3.1.5. Método `get_winner`

Determina el ganador del juego comparando las piedras en los mancalas.

```
def get_winner(self):
    if self.board[6] > self.board[13]:
        return "Jugador 1"
    elif self.board[6] < self.board[13]:
        return "Jugador 2"
    else:
        return "Empate"
```

## 3.2. Función principal

La función `main()` controla el flujo del juego.

```
def main():
    game = Mancala()
    while True:
        game.print_board()
        if game.player_turn == 1:
            pit = int(input("Jugador 1, elige un pozo (0-5): "))
        else:
            pit = int(input("Jugador 2, elige un pozo (7-12): "))

        if not game.make_move(pit):
            continue

        if game.check_game_over():
            game.print_board()
            print(" Juego terminado!")
            print("El ganador es:", game.get_winner())
            break
```

## 4. Interfaz por Consola

A continuación se presenta un ejemplo de salida en consola:

```
Tablero:
Jugador 2 (Mancala: 0)
Posicion fichas: 12  11  10  9  8  7
Pozo jugador 2:  [4, 4, 4, 4, 4, 4]
Pozo jugador 1  [4, 4, 4, 4, 4, 4]
Posicion fichas: 0  1  2  3  4  5
Jugador 1 (Mancala: 0)
Turno del Jugador 1
Jugador 1, elige un pozo (0-5):
```

### Descripción de la salida:

- Jugador 2 (Mancala: 0): Muestra la cantidad de piedras acumuladas por el Jugador 2.
- Pozo jugador 2: Representa el estado de los pozos del Jugador 2 (posiciones de la 12 a la 7).
- Pozo jugador 1: Representa el estado de los pozos del Jugador 1 (posiciones de la 0 a la 5).
- Jugador 1 (Mancala: 0): Muestra la cantidad de piedras acumuladas por el Jugador 1.
- Turno del Jugador 1: Indica a quién corresponde realizar el siguiente movimiento.
- Jugador 1, elige un pozo (0-5): Solicita al jugador que seleccione el pozo desde el cual desea mover.

```
Tablero:
Jugador 2 (Mancala: 0)
Posicion fichas: 12  11  10  9  8  7
Pozo jugador 2:  [4, 4, 4, 4, 4, 4]
Pozo jugador 1  [4, 4, 4, 4, 4, 4]
Posicion fichas: 0  1  2  3  4  5
Jugador 1 (Mancala: 0)
Turno del Jugador 1
Jugador 1, elige un pozo (0-5): █
```

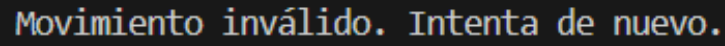
Figura 1: Vista del tablero del juego en consola

## 5. Errores Comunes

- Seleccionar un pozo fuera del rango permitido.
- Seleccionar un pozo que no contiene piedras.

En cualquiera de estos casos, el sistema mostrará el siguiente mensaje:

```
Movimiento invalido. Intenta de nuevo.
```



Movimiento inválido. Intenta de nuevo.

Figura 2: Ejemplo de mensaje por movimiento inválido

## Ejecución del Programa

Para ejecutar el juego, se debe contar con **Python 3** instalado en el sistema. Una vez ubicado en el directorio del archivo `mancala.py`, se debe ingresar el siguiente comando en la terminal o consola:

```
python mancala.py
```



python mancala.py

Figura 3: Ejemplo del comando para ejecutar el juego

## 6. Recomendaciones Generales

- Verificar que el archivo se encuentre correctamente nombrado como `mancala.py`.
- No cerrar la consola mientras la partida esté en curso.
- Puede reiniciar el juego simplemente ejecutando el archivo nuevamente.