# Partida de Póker

El objetivo de este ejercicio es implementar un **juego de póker** en modalidad "Texas Holdem". El juego se desarrolla entre dos jugadores y el que tenga la mejor mano gana.

La implementación del código se realizará utilizando *técnicas de programación orientada a objetos*.

## 1. Módulos

Se deben crear —al menos— los siguientes **módulos** y **clases**:

| Módulo | Explicación |
|---|---|
| `src/helpers.py` | Contiene funciones auxiliares para la implementación del juego. |
| `src/game.py` | Punto de entrada del proyecto. |
| `src/cards.py` | `Card` ⇒ Clase que representa una carta.<br>`Hand` ⇒ Clase que representa una mano. |
| `src/roles.py` | `Player` ⇒ Clase que representa un jugador. |

## 2. Requerimientos

### 2.1. Partida

El módulo `src/game.py` debe disponer de la siguiente función (*además de los métodos y atributos necesarios para el resto de su implementación*):

```python
def get_winner(
    players: list[Player],
    common_cards: list[Card],
    private_cards: list[list[Card]]
) -> tuple[Player | None, Hand]:
    ...
    """
    Devuelve el jugador que gana la partida de póker y la mano ganadora.

    En caso de empate, se devolverá None pero la mano ganadora
    sí tendrá un valor.

    players: Lista de jugadores.
    common_cards: Cartas comunes.
    private_cards: Cartas privadas de cada jugador.
    """
    # TODO
```


CC BY 4.0

## 2.2. Jugador

La clase `Player` debe disponer del siguiente constructor (*además de los métodos y atributos necesarios para el resto de su implementación*):

```python
class Player:
    def __init__(self, name: str):
        """
        Constructor de la clase Player.

        name: Nombre del jugador.
        """
        self.name = name
```

## 2.3. Carta

La clase `Card` debe disponer del siguiente constructor (*además de los métodos y atributos necesarios para el resto de su implementación*):

```python
class Card:
    def __init__(self, id: str):
        """
        Constructor de la clase Card.

        id: Identificador de la carta.
        """
        # TODO
```

Ejemplos de llamadas al constructor:
`Card('A♥') Card('K♠') Card('9♦') Card('4♣')`

## 2.4. Mano

La clase `Hand` debe disponer de lo siguiente (*además de los métodos y atributos necesarios para el resto de su implementación*):

- Un atributo `cards` que almacena una lista con las **5 cartas de la mano**.
- Un atributo `cat` que almacena **la categoría de la mano** (ver tabla de categorías).
- Un atributo `cat_rank` que almacena **el valor de la categoría** de la mano (ver tabla de categorías).
- El método `__contains__()` para determinar si una carta (`Card`) pertenece a la mano: `card in hand`


CC BY 4.0

### 2.4.1. Tabla de categorías

| `hand.cat` | **Explicación** | `hand.cat_rank` | **Explicación** |
|---|---|---|---|
| `Hand.HIGH_CARD` | Carta más alta | `'J'` | Carta más alta |
| `Hand.ONE_PAIR` | Pareja | `'5'` | Valor de la pareja |
| `Hand.TWO_PAIR` | Dobles parejas | `('10', '7')` | Valores de las parejas (Ordenado de mayor a menor) |
| `Hand.THREE_OF_A_KIND` | Trío | `'K'` | Valor del trío |
| `Hand.STRAIGHT` | Escalera* | `'9'` | Carta más alta de la escalera |
| `Hand.FLUSH` | Color | `'Q'` | Carta más alta del color |
| `Hand.FULL_HOUSE` | Trío y pareja | `('3', 'J')` | Valor del trío y de la pareja |
| `Hand.FOUR_OF_A_KIND` | Póker | `'Q'` | Valor del póker |
| `Hand.STRAIGHT_FLUSH` | Escalera de color* | `'7'` | Carta más alta de la escalera |

∗ *Aclaración de la escalera*: Aunque en el "póker verdadero" no es así, vamos a asumir (por simplicidad) que si hay un AS en la escalera, la única posibilidad es que sea la carta más alta `[10,J,Q,K,A]`. Es decir, el AS se mantiene como la carta de mayor valor.

### 2.4.2. Empates

Para que se produzca un empate la mejor mano de cada jugador debe "valer" exactamente lo mismo.

Ejemplo donde **no hay empate**:

| | |
|---|---|
| Jugador 1 | Q♦ 5♥ |
| Jugador 2 | J♣ 5♣ |
| Cartas comunes | A♠ 7♥ 7♠ 4♠ 2♠ |

- La mejor mano del jugador 1 es **pareja**: 7♥ 7♠ A♠ Q♦ 5♥
- La mejor mano del jugador 2 es **pareja**: 7♥ 7♠ A♠ J♣ 5♣

**Gana el jugador 1** porque, si bien ambos jugadores tienen pareja de 7, el jugador 1 tiene mejores cartas que el jugador 2.

Ejemplo donde **sí hay empate**:

| | |
|---|---|
| Jugador 1 | K♣ J♣ |
| Jugador 2 | J♦ 8♥ |
| Cartas comunes | A♣ J♥ 9♠ 9♥ 8♦ |

- La mejor mano del jug1 es **dobles parejas**: J♣ J♦ 9♠ 9♥ A♣
- La mejor mano del jug2 es **dobles parejas**: J♣ J♦ 9♠ 9♥ A♣

**Hay empate** porque ambos jugadores tienen la misma "mejor mano" y, además, el resto de "mejores cartas" son las mismas.

# 3. Módulo "helpers"

El módulo `src/helpers.py` contiene funciones de apoyo al desarrollo del proyecto.

La más importante es `combinations(values, n)` que genera todas las combinaciones del **iterable** `values` con tamaño $n$.

Un ejemplo:

```
1 >>> list(helpers.combinations((1, 2, 3, 4, 5), n=3))
```


CC BY 4.0

```
[(1, 2, 3),
 (1, 2, 4),
 (1, 2, 5),
 (1, 3, 4),
 (1, 3, 5),
 (1, 4, 5),
 (2, 3, 4),
 (2, 3, 5),
 (2, 4, 5),
 (3, 4, 5)]
```

Notas:

- Lo que le pasamos a esta función es un **iterable** con lo que se podría pasar también una lista de objetos de tipo `Card`.
- El parámetro `n` debe pasarse como un *parámetro nominal*.

# 4. Referencias

- Anatomía de una carta de póker.
- Lista de posibles manos ganadoras.
- Calculadora de manos ganadoras.


CC BY 4.0