PL0b: El juego de la Mona

Preparación del juego:

Antes de empezar a repartir, se aparta una carta cualquiera de la baraja sin que ningún jugador la vea. Después de esta partida se deberá cambiar esta carta por otra al azar para que no se sepa cuál es. Se reparten las cartas restantes entre todos los jugadores, empieza el turno el de la derecha del repartidor. La partida sigue en el sentido contrario a las agujas del reloj. En este juego también jugará la máquina, siendo uno de los jugadores.

Objetivo del juego:

Para ganar deberás ser el primer jugador en deshacerse de todas las cartas. En este juego es más relevante quién pierde en lugar de quién gana. El que se quede con la última carta desparejada (“mona”) va a ser el perdedor.

Normas del juego:

En la primera fase del juego, los jugadores se descartan de todas las parejas que tengan por turnos. Las parejas se forman con dos cartas del mismo valor y los palos, **bastos con espadas** y **copas con oros**. Para deshacerte de las parejas, tienes que dejarlas boca arriba en la mesa para que todos los jugadores vean que no haces trampas. No importa que un jugador no se de cuenta o se olvide de descartarse, ya que el único perjudicado va a ser él mismo. Cuando haya terminado la primera ronda de descarte, comienza la segunda fase.

En esta parte del juego, comienza el jugador que se haya quedado con más cartas en su mano. En caso de empate empezará el que esté más cerca del repartidor en sentido antihorario. El jugador que empiece esta ronda, ofrecerá sus cartas boca abajo al jugador de su derecha, éste deberá elegir una de ellas y juntarla con sus cartas. En el caso de que pueda formar alguna pareja, desechará estas cartas boca arriba y después ofrecerá sus cartas cara abajo al jugador de su derecha. Si con la carta que ha robado no consigue formar ninguna pareja, entonces deberá ofrecer sus cartas directamente al siguiente jugador. Esta mecánica se repetirá hasta que todos los jugadores se descarten de sus cartas y solo quede un jugador con la carta desparejada. Este jugador será la mona (perdedor)

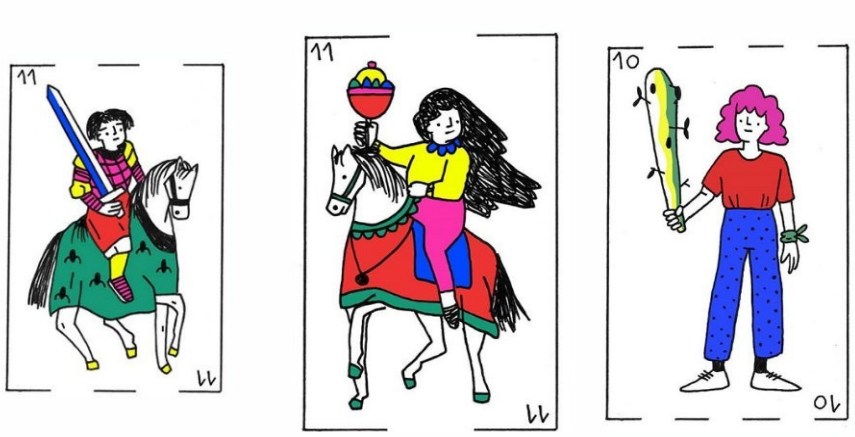
Objetivo:

El objetivo de esta práctica es seguir adquiriendo conocimiento sobre el lenguaje Python. En este caso, se pretende aprender el uso de la herencia en el lenguaje Python.

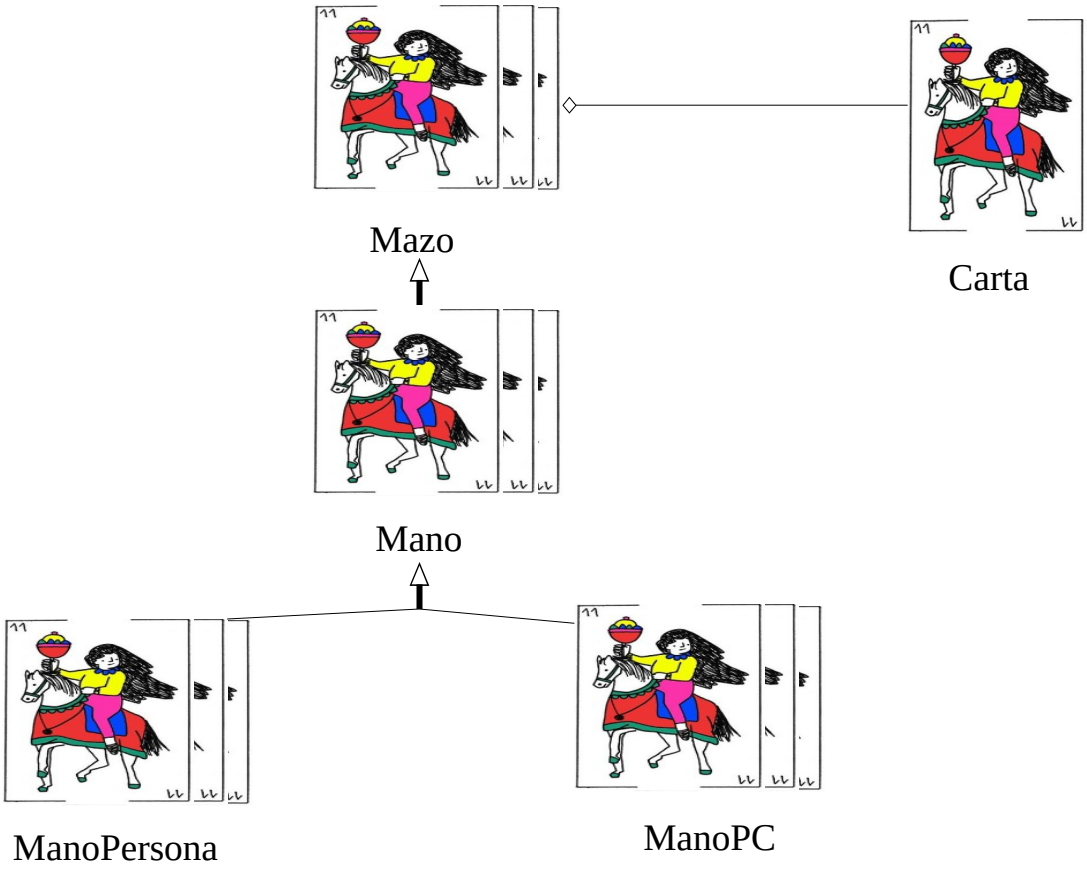
Para ello, se deberá implementar el juego de la Mona. En el fichero “Mona.zip” encontrareis varios ficheros incompletos que deberéis de terminar de implementar.

A continuación se muestran los ficheros con los subprogramas incompletos.

Herencia:



Clases del juego:



Carta: atributos y métodos

Atributos:

- palo
- rango

```
class Carta(object):
 palo_nombres = ["Bastos", "Oros", "Copas", "Espadas"]
      rango_nombres = [None, "As", "2", "3", "4", "5", "6", "7", "Sota", "Caballo",
"Rey"]

    def __init__(self, palo=0, rango=2):
        self.palo = palo
        self.rango = rango
```

Métodos: sobreescribir str, cmp, eq y ne (que se pueden emplear en ordenarAsc o en métodos internos)

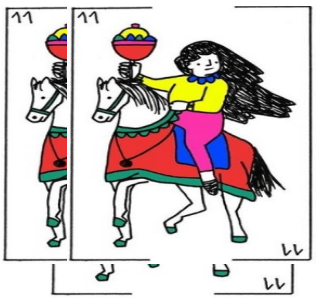
```
def __str__(self):
        return '%s of %s' % (Carta.rango_nombres[self.rango],
                Carta.palo_nombres[self.palo])

def __cmp__(self, other):
        """Compara una carta con otra
        Devuelve un positivo si self > other; negativo si other > self; y 0 si
        equivalentes"""
        #TODO Añadir el codigo para reescribir el cmp
        rdo = NotImplemented
        return rdo

def __eq__(self, other):
        rdo = NotImplemented
        if isinstance(other, Carta):
            #TODO Añadir el codigo para reescribir el eq
        return rdo

def __ne__(self, other):
        """self != other"""
        eq = Carta.__eq__(self, other)
        return not eq
```

Mazo: atributos y métodos



Atributos:

- cartas=[]

```
class Mazo (object):

def __init__(self):
        self.cartas = [ ]
        for palo in range(4):
                for rango in range(1, 11):
                        carta = Carta(palo, rango)
                        self.cartas.append(carta)

def __str__(self):
        res = [ ]
        for carta in self.cartas:
                res.append(str(carta))
        return '\n'.join(res)


def anadir_carta(self, carta):
"""Añade una carta al mazo."""

def eliminar_carta(self, carta):
"""Elimina una carta del mazo."""

def esta_carta(self, carta):
"""Comprueba si una carta está en el mazo. True si esta, False si no"""

def pop_carta(self, i=-1):
"""Saca una carta del mazo.
        i: el índice de la carta a sacar (por defecto -1, es decir, la ultima); """

def barajar(self):
"""Baraja las cartas del mazo."""

def ordenarAsc(self):
"""Ordena las cartas del mazo en orden ascendente. Puede venir bien para ordenar las
cartas de la mano para buscar las posibles parejas"""

def mover_cartas(self, mano, num):
"""Mueve num cartas desde el mazo a la mano.
        mano: objeto destino perteneciente a la clase Mano
        num: numero de cartas a desplazar """
```

```
def imprimir(self):
"""Imprime las cartas contenidas en el mazo”””

def esta_vacio(self):
"""True si la lista de cartas esta vacía."""

def repartir_cartas(self, manos, num_cartas=999):
        num_manos = len(manos)
        i=0
        while i <= num_cartas and not self.esta_vacio():
            #TODO Anadir el codigo          # coger la carta de la cima
            mano = manos[i % num_manos]  # calcular a quien repartir
            mano.anadir_carta(carta)          # anadir una carta a una mano
            i+=1
```

Mano: atributos y métodos

Atributos:

- los que hereda (cartas)
- jugador

```
def __init__(self, jugador=''):
        self.cartas = [ ]
        self.jugador = jugador

def imprimir(self):
        toPrint ='la mano de {0} '.format(self.jugador)
        #### TODO
```

ManoPersona: atributos y métodos

Atributos:

- los que hereda (mano)

```
#####################################
   # Forman pareja:
   # numX de bastos y numX de espadas
   # numX de copas  y numx de oros
#####################################
def eliminar_parejas(self):
        cont = 0
        salir = False # para salir cuando la pareja que se pretende eliminar no sea tal
        while not salir:
                super().imprimir()
                print("Que cartas forman pareja? de 0 a {0}".format(len(self.cartas)-1))
                idxCarta,idxPareja = map(int, input().split(','))
                carta = self.cartas[idxCarta]
                posiblePareja = self.cartas[idxPareja]
                pareja = Carta(3 - carta.palo, carta.rango)
                if #TODO comprobar si la que debiera ser pareja es la esperada y eliminar ambas de la mano
                        print("En la mano de {0} forman par {1}
                                {2}".format(self.jugador,carta,pareja))
                        cont += 1
                else:
                        salir = True
         return cont
```

ManoPC: atributos y métodos

Atributos:

- los que hereda (mano)

```
#####################################
   # Forman pareja:
   # numX de bastos y numX de espadas
   # numX de copas  y numx de oros
#####################################
def eliminar_parejas(self):
        cont = 0
        #TODO elimina las parejas AUTOMATICAMENTE
        return cont
```