Volem implementar una versió simplificada del joc de la Oca. Per controlar el funcionament de tot el joc tindrem una classe `Tauler`, amb la declaració i constructor que us mostrem a continuació:

```
class Tauler():
    def __init__(self):
        self._caselles = []
        self._jugadors = []
        self._jugador_actual = 0
```

Aquesta classe té aquests atributs:
- `_caselles`: és una llista amb objectes de la classe `Casella` que us expliquem més endavant per guardar la informació de totes les caselles del tauler del joc.
- `_jugadors`: és una llista amb objectes de la classe `Jugador` que us expliquem més endavant per guardar l'estat actual de cadascun dels jugadors de la partida.
- `_jugador_actual`: és un enter que guarda l'índex de la llista de jugadors que correspon al jugador que li toca tirar en el torn actual.

Exercici 1

La **classe `Jugador`** guardarà tota la informació necessària de cada jugador pel desenvolupament del joc. Us donem ja feta la declaració de la classe i alguns mètodes:

```
class Jugador:
    def __init__(self):
        self._casella = 1
        self._pot_tirar = True
        self._n_torns_inactiu = 0
        self._guanyador = False

    def posicio(self):
        return self._casella

    def mou(self, casella):
        self._casella = casella

    def guanya(self):
        self._guanyador = True
```

A nivell d'atributs, la classe té:
- Un atribut `_casella` de tipus enter per guardar la posició del tauler on està en cada moment el jugador.
- Un atribut booleà `_pot_tirar` que indica si el jugador pot tirar en el torn actual, o està atrapat en alguna casella que implica estar alguns torns sense tirar.
- Un atribut `n_torns_inactiu` que indica, si el jugador està atrapat en alguna casella que no li deixa tirar, quants torns han de passar per poder tornar a tirar.
- Un atribut booleà `_guanyador` que només es posarà a `True` quan el jugador arribi a la casella final del joc.

A nivell de mètodes, apart del constructor, us donem fets aquests tres mètodes:
- `posicio()`: retorna el número de casella on està el jugador.
- `mou(casella)`: aquest mètode es cridarà quan el jugador es mogui de casella. Serveix per canviar la posició del tauler on està el jugador.
- `guanya()`: retorna si el jugador ha guanyat la partida o no.

Us demanem completar la definició de la classe `Jugador` afegint aquests dos mètodes:
- **`set_inactiu(n_torns)`**: aquest mètode es cridarà cada cop que el jugador caigui en una casella que impliqui quedar-se uns quants torns sense poder tirar. Ha de canviar l'estat del jugador per indicar que en el següent torn no pot tirar i per guardar quants torns haurà d'estar el jugador sense poder tirar.
- **`pot_tirar()`**: aquest mètode ens retornarà si el jugador pot tirar en el torn actual, o no ho pot fer perquè està atrapat en alguna casella especial. En aquest cas, a més a més, haurà de decrementar el número de torns que li queden per poder tornar a tirar. Quan el número de torns per tirar arribi a zero haurem de canviar l'estat per indicar que podem tornar a tirar.

Exercici 2

La **classe `Casella`** guarda la informació bàsica de cada casella del joc i permet gestionar les accions que s'han de fer quan el jugador arriba a la casella, en funció del tipus de casella.

Us donem ja feta la implementació de la classe `Casella` en el cas que sigui una casella no especial, és a dir, una casella en la que no passa res si s'hi cau (ni es perd el torn, ni es salta a una altra casella, ...).

```
class Casella:
    def __init__(self, posicio, tauler):
        self._posicio = posicio
        self._tauler = tauler

    def entra_jugador(self, jugador):
        jugador.mou(self._posicio)
        return False

    @property
    def posicio(self):
        return self._posicio

    def es_oca(self):
        return False
```

Aquesta classe `Casella` té com a atributs un enter que indica a quina posició del tauler correspon la casella i una referència a l'objecte de la classe `Tauler`, que haurem d'utilitzar per fer alguna de les accions associades a certes caselles.

Com a mètodes, apart de la propietat per retornar el valor de la posició de la casella, té el mètode `es_oca` que retorna si la casella correspon a una oca o no, i el mètode `entra_jugador` que es cridarà cada vegada que un jugador arriba a la casella i que serveix per modificar l'estat del jugador en funció de l'acció associada a la casella. En el cas de les caselles no especials, l'únic que es fa és cridar al mètode `mou` de la classe `Jugador`, que modificarà la posició a la que està el jugador que es passa com a paràmetre. Aquest mètode retornarà `True` si després de caure en aquesta casella el jugador conserva el torn (per exemple, quan cau a una oca). En el cas de les caselles no especials retornem `False` perquè el jugador no conserva el torn.

Us demanem que a partir d'aquesta definició de la classe `Casella`, **declareu utilitzant herència** altres **classes derivades** que permetin **tractar algunes de les caselles especials del joc de la Oca**. En aquesta versió simplificada només considerarem aquestes caselles especials:

- **Oca**: quan el jugador arriba a la casella, salta fins a la següent casella marcada com a Oca i manté el torn.
- **Pou**: el jugador s'ha de quedar dos torns sense jugar.
- **Mort**: el jugador torna a la casella inicial.
- **Final**: casella final del joc. Si el jugador hi arriba guanya la partida.

Exercici 3

La **classe `Tauler`** guarda tota la informació del joc i té els mètodes necessaris per gestionar els torns dels jugadors. Us donem ja fet el mètode `torn_joc` que és el mètode que es cridarà cada cop que un jugador tiri el dau:

```
def torn_joc(self, valor_dau):
   jugador_torn = self._jugadors[self._jugador_actual]
    torna_tirar = False
    if jugador_torn.pot_tirar():
      nova_posicio = jugador_torn.posicio() + valor_dau
        if nova_posicio <= len(self._caselles):
          torna_tirar =
          self._caselles[nova_posicio-1].entra_jugador(jugador_torn)
   if not torna_tirar:
      self._jugador_actual =
          (self._jugador_actual  + 1) % len(self._jugadors)
```

Aquest mètode rep com a paràmetre el valor que ha sortit al dau. Si el jugador que té el torn no pot tirar, no es fa res, simplement es salta el torn. Si el jugador actual pot tirar es calcula la nova posició a la que ha d'anar. Si la posició és més gran que la casella final tampoc es fa res, considerem que el jugador no es pot moure. Si es pot moure, es crida al mètode `entra_jugador` de la casella on arriba perquè es faci l'acció corresponent a aquella casella (oca, pou, mort, final o casella no especial) segons s'ha explicat a l'exercici

anterior. Com hem explicat també abans aquest mètode `entra_jugador` retornarà `True` o `False` en funció de si el jugador conserva o no el torn. Si el jugador no conserva el torn es passa el torn al següent jugador modificant el valor de l'atribut `_jugador_actual`

Us demanem que implementeu aquests dos mètodes de la classe `Tauler`:

- `inicialitza(nom_fitxer, n_jugadors)`: aquest mètode ha d'inicialitzar les caselles del tauler, creant els objectes corresponents de la classe Casella o d'alguna de les seves classes derivades, segons el tipus de la casella, llegint del fitxer que es passa com a paràmetre les dades de totes les caselles del joc. També ha de fixar i inicialitzar el nº de jugadors al valor que es passa com a paràmetre. El fitxer amb les dades de les caselles tindrà aquest format:

  ```
  POSICIO_1 TIPUS_1
  POSICIO_2 TIPUS_1
  ...
  POSICIO_N TIPUS_N
  ```

  `POSICIO` és un enter indicant el nº de la casella al tauler. Podeu suposar que les caselles estan ordenades al fitxer per la seva posició, des del 1 fins al nº total de caselles.
  `TIPUS` és un caràcter que indica el tipus de casella a cada posició segons la codificació següent: 'O' – oca, 'P' – Pou, 'M' – Mort, 'F' – Final, 'N' – Casella no especial.

- `salta_oca(jugador)`: aquest mètode es cridarà quan un jugador caigui en una oca. Ha de buscar dins del tauler la següent oca i fer que el jugador que es passa com a paràmetre es mogui a la casella corresponent.

Notes:

- Us donem un programa de test per poder verificar que el funcionament del codi que feu és correcte.
- Tot i que valorarem que el programa de test funcioni correctament, tingueu en compte que aquest no serà el criteri principal d'avaluació, sinó que sigui conceptualment correcte i s'ajusti als que hem explicat a classe.
- Podeu afegir tot el que creieu que sigui necessari (atributs, mètodes) a les classes que s'indiquen, si penseu que us ajuda en la implementació del que es demana.
- Podeu afegir totes les instruccions `print` que creieu necessàries perquè el programa vagi donant informació del que fa i poder comprovar si el seu funcionament és correcte.