Partie 2

1. La classe Joueur

Afin de représenter les joueurs, on définit la classe `Joueur`. Cette classe contiendra les 5 attributs suivants : `prenom, argent, mise, etat, paquetJoueur`.
`prenom` : prénom du joueur
`argent` : argent dont dispose le joueur
`mise` : argent misé par le joueur en début de partie
`etat` : état dans lequel se trouve le joueur. Les états sont les suivants : 'run', 'stop', 'perdu', 'inactif', 'gagne'.
`paquetJoueur` : Paquet des cartes du joueur

Le rôle de l'attribut `etat` est le suivant : En début de partie un joueur est dans l'état 'run' s'il a assez d'argent pour jouer, sinon il est mis dans l'état 'inactif'. Si le joueur décide d'arrêter il devra être mis dans l'état 'stop'. S'il gagne ou s'il perd il devra être mis respectivement dans les états 'gagne' ou 'perdu'

Lors de la création d'un joueur on devra passer en argument le prénom du joueur et l'argent dont il dispose. L'attribut `mise` sera initialisé à 0 et l'attribut `etat` à `None`. On devra également créer l'attribut `paquetJoueur` comme étant un paquet de carte vide.
Les méthodes suivantes devront être implémentées (plus éventuellement des getters et setters) :
`__str__` : doit permettre l'affichage du prenom, de la somme d'argent dont il dispose et de son état
`setEtatJoueur(nouvelEtat)` : modifie l'état du joueur. Seuls les états 'run', 'stop', 'perdu', 'inactif', 'gagne' sont autorisés
`ajouterCarteDansPaquetJoueur (carte)` : ajoute la carte dans le paquet du joueur
`addArgentJoueur(Argent)` : ajouter la somme Argent à l'attribut argent
`ClearPaquetJoueur()` : supprime toute les cartes du paquet du joueur
`miseJoueur(miseMin)` : lit au clavier la somme que souhaite miser le joueur et la retourne. Attention cette somme doit être supérieur ou égale à `miseMin`. De plus la mise doit être compatible avec la somme dont dispose le joueur (attribut `argent`)
`action()` : lit au clavier l'action que souhaite faire le joueur et la retourne. Seule les 4 actions '1', '2' ,'p', 's' sont possibles (tirer 1 ou 2 cartes, p : passer son tour, s : arrêter)

Le programme de test suivant devra afficher :

```
if __name__ == '__main__':
    j1=Joueur('Obélix',100)
    j1.ajouterCarteDansPaquetJoueur(Carte('as','pique',1))
    j1.ajouterCarteDansPaquetJoueur(Carte('7','coeur',7))
    j1.setEtatJoueur('run')

    j2=Joueur('Martine',120)
    j2.ajouterCarteDansPaquetJoueur(Carte('as','coeur',1))
    j2.ajouterCarteDansPaquetJoueur(Carte('roi','carreau',10))
    print(j1,j2)
```

```
Obélix : 100 Euros Etat : run
Martine : 120 Euros Etat : None
```

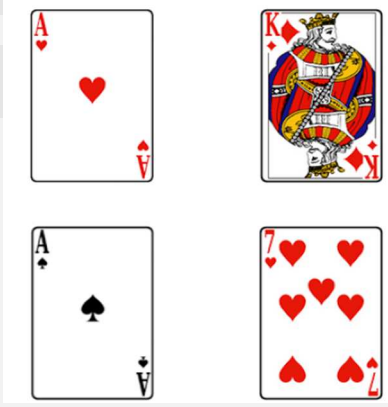
On ajoute la méthode plot afin d'afficher les cartes d'un joueur :

```
def plot(self,fig=None,left=0,bottom=0,width=0.2,height=0.2,shift=0.05):
        self.paquetJoueur.plot(fig,left,bottom,width,height,shift)
```

On ajoute à la fin du programme principal les 3 lignes suivantes :

```
    fig=plt.figure()
    j1.plot(fig,bottom=0.0,width=0.4,height=0.4,shift=0.4)
    j2.plot(fig,bottom=0.5,width=0.4,height=0.4,shift=0.4)
```

Vérifier que vous obtenez l'affichage ci-contre



2. La classe Partie

On définit la classe `Partie` permet de définir des éléments généraux d'une partie de carte.
Cette classe contient 3 attributs : `nomDuJeu`, `listeJoueurs`, `miseMin`
`nomDuJeu` : contient le nom du jeu
`listeJoueurs` : liste des joueurs
`miseMin` : somme minimale pouvant être misée

Le constructeur devra initialiser ces 3 attributs à partir des 3 arguments puis mettre à jour l'état des joueurs grâce à la méthode `setEtatJoueurs()`, puis supprimer les cartes des paquets des joueurs (au cas où il y a eu une partie précédente) à l'aide de

la méthode `ClearPaquetJoueurs()`. Les méthodes suivantes (plus éventuellement des getters et setters) devront être implémentées :
`__str__()`
`setEtatJoueurs()` : pour chaque joueurs de la liste cette méthode doit initialiser l'état du joueur soit 'run' si la somme dont il dispose est supérieure à miseMin, soit à l'état 'inactif' si il ne possède pas suffisamment d'argent.
`ClearPaquetJoueurs()` : supprime les cartes des paquets de cartes des joueurs

Le programme suivant devra afficher :

```
if __name__ == '__main__':
    joueurs=[Joueur('Obélix',80),Joueur('Sarah',200)]
    P=Partie('Black JAck',joueurs,100)
    print(P)
```

```
Jeu de Black JAck
Il y a 2 joueurs
Obélix : 80 Euros Etat : inactif
Sarah : 200 Euros Etat : run
```

3. La classe PartieBlackJack

Cette classe définit les règles du jeu. Elle hérite de la classe Partie. Pour définir une partie de Black Jack il conviendra de passer en argument les informations suivantes : la liste des joueurs, la mise minimale, la somme d'argent dont dispose la banque, le nombre N de jeux de 52 cartes utilisé.
Son constructeur devra créer 3 attributs supplémentaires :
`banque` : somme d'argent de la banque disponible
`paquetCroupier` : jeu de cartes Black Jack composé de N jeux de 52 cartes
`fig` : utilisé pour l'affichage (self.fig = plt.figure(figsize=(4,3),dpi=120))

Les méthodes suivantes devront être implémentées :
`FinDePartie()` : Cette méthode devra retourner True si la partie est terminée sinon elle retournera False. Une partie est terminée si au moins 1 joueurs est dans l'état 'gagne'. Une partie n'est pas terminée si au moins 1 joueur est dans l'état 'run'

`MiseAJourArgent()` : cette méthode sera appelée en fin de partie pour mettre à jour l'argent des joueurs ainsi que celui de la banque selon les règles suivantes :
Un joueur qui a gagné reçoit 1,5 fois sa mise de la banque
Un joueur qui a perdu donne toute sa mise à la banque
Un joueur qui a arrêté verse 0,5 fois sa mise à la banque

`plot()` : affiche les cartes de tout les joueurs :

```
def plot(self):
        for i,joueur in enumerate(self.listeJoueurs):
            joueur.plot(self.fig,left=0.0,bottom=i*0.25,width=0.2,height=0.2,shift=0.05)
        plt.pause(0.1)
```

`run()` : c'est la méthode dans laquelle sera implémentée la règle du jeu. L'algorithme général est le suivant :

- On commencera par supprimer les cartes des paquets des joueurs et mettre à jour l'état des joueurs
- Pour chaque joueur :
  - Si le joueur est dans l'état 'run' alors
    - Tirer une carte du jeu de carte et la mettre dans le paquet du joueur
- Afficher les cartes des joueurs
- Tant que la partie n'est pas finie faire :
  - Pour chaque joueur faire :
    - SI le joueur est dans l'état 'run' alors :
      - A = action du joueur
      - Si A == 's' mettre l'état du joueur à 'stop'
      - Si A == '1' ou '2' alors :
        - tirer 1 ou 2 cartes du jeu de carte et les mettre dans le paquet du joueur
        - Calculer le total des points du joueur
        - Si le total vaut 21 mettre l'état du joueur à 'gagne'
        - Si le total > 21 mettre l'tat du joueur à 'perdu'
- Mettre à jour les sommes d'argent des joueurs et de la banque
- Afficher les sommes d'argent des joueurs et de la banque

Le programme principal de test est le suivant :

```
if __name__ =='__main__':
    j1=Joueur('Obélix',100)
    j2=Joueur('Nathalie',120)
    UnePartie = PartieBlackJack([j1,j2],20,1000,1)
    UnePartie.run()
```

========== FIN DE LA PARTIE 2 ==========