Documentation du fichier `joueur.py`

Introduction

Ce fichier contient la classe `Joueur`, qui gère les actions et l'état d'un joueur dans un jeu de Monopoly. Les actions principales incluent le tirage des dés, le déplacement sur le plateau, l'achat de terrains et le paiement de loyers.

Classe `Joueur`

Description

La classe `Joueur` représente un joueur dans le jeu Monopoly. Chaque joueur possède un nom, un solde d'argent, une position sur le plateau, et plusieurs états qui peuvent influencer son comportement, comme être en prison, pouvoir rejouer, ou attendre plusieurs tours.

Constructeur

- `__init__(self, nom, solde=2000, position=0)`
  - **Paramètres** :
    - `nom` (str) : Le nom du joueur.
    - `solde` (int) : Le solde d'argent initial du joueur. Par défaut, il est de 2000€.
    - `position` (int) : La position initiale du joueur sur le plateau. Par défaut, il commence à la case 0.
  - **Effet** : Initialise un joueur avec les attributs nécessaires (nom, solde, position, etc.).

Méthodes

- `tirer_de(self)`
  - Lance un dé et retourne le résultat (un nombre aléatoire entre 1 et 6).
  - **Retour** : Un entier (résultat du lancé du dé).

- `deplacement(self, nb_cases)`
  - Déplace le joueur sur le plateau d'un nombre de cases spécifié. Si le joueur a un effet actif (`lance_divise_par_2`), le déplacement est divisé par 2.
  - **Paramètres** :
    - `nb_cases` (int) : Le nombre de cases à déplacer.
  - **Effet** : Met à jour la position du joueur et détermine s'il a effectué un tour complet.

- `acheter(self, terrain)`
  - Permet au joueur d'acheter un terrain s'il en a les moyens.
  - **Paramètres** :
    - `terrain` (Terrain) : L'instance de terrain que le joueur souhaite acheter.

    - **Effet** : Si le joueur accepte l'achat et possède assez d'argent, le terrain est acheté et ajouté à sa liste de propriétés.

- `payer(self, terrain)`
    - Le joueur paie le loyer du terrain sur lequel il se trouve, s'il a un propriétaire.
    - **Paramètres** :
        - `terrain` (Terrain) : L'instance du terrain pour lequel le loyer doit être payé.
    - **Effet** : Si le terrain a un propriétaire, le joueur lui verse le loyer. Si le joueur n'a pas assez d'argent, il lui donne tout son solde.

Attributs

- `nom` (str) : Le nom du joueur.
- `solde` (int) : Le solde d'argent du joueur.
- `position` (int) : La position du joueur sur le plateau.
- `terter` (list) : Liste des terrains possédés par le joueur.
- `en_prison` (bool) : Indique si le joueur est en prison.
- `lance_divise_par_2` (bool) : Indique si le prochain lancé du joueur doit être divisé par 2.
- `tours_a_attendre` (int) : Nombre de tours que le joueur doit attendre (par exemple, s'il est en prison).
- `peut_rejouer` (bool) : Indique si le joueur peut rejouer immédiatement après une carte Chance.
- `a_fait_tour_complet` (bool) : Indique si le joueur a effectué un tour complet sur le plateau (passé la case 23).

Exemple d'utilisation

```
# Création d'un joueur
joueur1 = Joueur("Alice")

# Lancer de dé
de_resultat = joueur1.tirer_de()
print(f"{joueur1.nom} a tiré un {de_resultat}")

# Déplacement du joueur
joueur1.deplacement(de_resultat)
print(f"{joueur1.nom} est maintenant à la case {joueur1.position}")

# Acheter un terrain
terrain = Terrain("Avenue des Champs-Élysées", 300)
joueur1.acheter(terrain)

# Payer le loyer
joueur1.payer(terrain)
```

Classe `Terrain`

Description

La classe `Terrain` représente un terrain dans le jeu de Monopoly. Chaque terrain possède des informations telles que son nom, son numéro de salle, son propriétaire, le nombre de maisons et d'hôtels présents sur ce terrain. Elle offre également des méthodes permettant d'acheter le terrain, de calculer le loyer et d'ajouter des maisons ou des hôtels.

Attributs

- `cout_achat` (dict) : Dictionnaire qui contient le coût d'achat pour chaque numéro de salle.
- `loyer_base` (dict) : Dictionnaire qui contient le loyer de base pour chaque numéro de salle.
- `cout_maisons` (dict) : Dictionnaire qui contient le coût pour ajouter une maison pour chaque numéro de salle.
- `cout_hotels` (dict) : Dictionnaire qui contient le coût pour ajouter un hôtel pour chaque numéro de salle.
- `nom` (str) : Nom du terrain (hérité de la classe `Case`).
- `salle` (str) : Numéro de salle du terrain (par exemple, "T11", "T24 BIS").
- `proprio` (Joueur) : Propriétaire du terrain (peut être `None` si le terrain n'est pas encore acheté).
- `nbr_maisons` (int) : Nombre de maisons présentes sur le terrain.
- `nbr_hotels` (int) : Nombre d'hôtels présents sur le terrain.
- `prix` (int) : Prix d'achat du terrain, récupéré à partir du dictionnaire `cout_achat`.
- `loyer` (int) : Loyer de base du terrain, récupéré à partir du dictionnaire `loyer_base`.
- `cout_maison` (int) : Coût pour ajouter une maison, récupéré à partir du dictionnaire `cout_maisons`.
- `cout_hotel` (int) : Coût pour ajouter un hôtel, récupéré à partir du dictionnaire `cout_hotels`.

Méthodes

- `__init__(self, nom, salle, proprio=None)`
  - **Paramètres** :
    - `nom` (str) : Nom du terrain.
    - `salle` (str) : Numéro de salle du terrain (par exemple, "T11").
    - `proprio` (Joueur ou None) : Propriétaire du terrain, par défaut `None`.
  - **Effet** : Initialise un terrain avec le nom, le numéro de salle, et le propriétaire (s'il existe). Définit également les valeurs associées au terrain comme le prix, le loyer, le coût des maisons et des hôtels en fonction du numéro de salle.
- `est_achetable(self)`
  - **Retourne** : `True` si le terrain n'a pas de propriétaire, sinon `False`.
  - **Effet** : Vérifie si le terrain peut être acheté (c'est-à-dire si son propriétaire est `None`).
- `ameliorer_terrain(self, joueur)`

  - **Paramètres** :
    - `joueur` (Joueur) : Le joueur qui souhaite améliorer le terrain.
  - **Effet** : Permet à un joueur d'ajouter une maison ou de transformer 4 maisons en 1 hôtel sur le terrain, si le joueur a assez d'argent. Si un hôtel est déjà présent, l'amélioration est impossible.

- `acheter_terrain(self, joueur)`

  - **Paramètres** :
    - `joueur` (Joueur) : Le joueur qui souhaite acheter le terrain.
  - **Effet** : Permet à un joueur d'acheter le terrain si celui-ci est libre et si le joueur a suffisamment d'argent. Le propriétaire du terrain est mis à jour et l'argent du joueur est déduit.

- `get_loyer(self)`

  - **Retourne** : Le loyer à payer pour ce terrain en fonction du nombre de maisons ou d'hôtels présents.
  - **Effet** : Si le terrain a un hôtel, le loyer est multiplié par 10. Si le terrain a des maisons, le loyer est augmenté de 50% par maison. Sinon, le loyer de base est utilisé.

Exemple d'utilisation

```
# Création d'un joueur
joueur = Joueur(nom="Alice")

# Création d'un terrain
terrain = Terrain(nom="Sunny head beach", salle="T11")

# Achat du terrain
terrain.acheter_terrain(joueur)

# Ajouter une maison
terrain.ameliorer_terrain(joueur)

# Récupérer le loyer du terrain
loyer = terrain.get_loyer()
print(f"Le loyer du terrain {terrain.nom} est de {loyer}€.")
```

Classe `Case`

Description

Représente une case dans le jeu de Monopoly, avec un nom unique.

Méthodes

- `__init__(self, nom)` : Initialise la case avec un nom.
  - `nom` (str) : Le nom de la case.

- `__str__(self)` : Retourne le nom de la case sous forme de chaîne de caractères.

Classe `CaseSpeciale`

Description

La classe `CaseSpeciale` étend la classe `Case` et représente une case spéciale dans le jeu Monopoly. Elle peut être de type "chance", "bonus", "malus" ou spécifiquement "prison". Elle gère des effets particuliers pour les joueurs lorsqu'ils y atterrissent.

Constructeur

- `__init__(self, nom, type_case)`
  - **Paramètres** :
    - `nom` (str) : Le nom de la case (hérité de la classe `Case`).
    - `type_case` (str) : Le type de la case, qui peut être "bonus", "malus", ou "chance". Si le type est "chance", une série de cartes Chance est générée.

Méthodes

- `aller_prison(self, joueur)`
  - Envoie un joueur en prison si la case est "En Prison".
  - **Paramètre** : `joueur` (Joueur) : Le joueur à envoyer en prison.
  - **Effet** : Met à jour la position du joueur à la case de la prison et initialise les variables de prison.
- `gerer_prison(self, joueur)`
  - Gère le cas où un joueur est en prison.
  - **Paramètre** : `joueur` (Joueur) : Le joueur en prison.
  - **Effet** : Permet au joueur de sortir de prison soit en lançant un dé (faire un 6) soit en payant une somme pour sortir immédiatement (50€).
  - Après 3 tours, le joueur est libéré automatiquement.

Attributs

- `tabCartes` (list) : Liste des cartes Chance disponibles lorsque la case est de type "chance". Chaque carte est un objet de type `Chance`.

Classe `Chance`

Description

La classe `Chance` représente une carte Chance dans le jeu Monopoly. Chaque carte a un effet particulier qui peut modifier le jeu de différentes manières, comme avancer de cases, gagner ou perdre de l'argent, ou encore provoquer des actions spéciales.

Constructeur

- `__init__(self, nom)`
  - **Paramètre** :

        - `nom` (str) : Le nom de la carte Chance (par exemple, "Récréation", "Bonne Note", etc.).
    - **Effet** : Initialise la carte avec un effet spécifique basé sur le nom. Chaque carte a un texte explicatif (`texte`) et parfois un déplacement (`deplacement`) ou une autre valeur associée à l'effet.

Méthodes

- `appliquer_effet(self, joueur, partie)`
    - Applique l'effet de la carte Chance au joueur. En fonction de la carte tirée, l'effet peut inclure :
        - Avancer ou reculer de certaines cases.
        - Gagner ou perdre de l'argent.
        - Bloquer le joueur pendant un certain nombre de tours.
        - Modifier certaines propriétés du joueur, comme la possibilité de rejouer ou la division de son prochain lancé de dé.
    - **Paramètres** :
        - `joueur` (Joueur) : Le joueur auquel appliquer l'effet de la carte.
        - `partie` (Partie) : L'instance de la partie qui permet de gérer les déplacements et autres effets du jeu.
    - **Effet** : Modifie l'état du joueur et/ou de la partie en fonction de la carte tirée.

Attributs

- `nom` (str) : Le nom de la carte Chance (ex : "Récréation", "Journée banalisée", etc.).
- `deplacement` (int) : Le nombre de cases à déplacer le joueur (s'il y en a un).
- `texte` (str) : Le texte explicatif de l'effet de la carte.
- `hasard` (int, optionnel) : Un nombre aléatoire généré pour certaines cartes qui utilisent le hasard (ex : "Contrôle surprise").

Exemples d'effets de cartes :

- **Récréation** : Avance le joueur de 1 à 6 cases.
- **Journée banalisée** : Permet au joueur de rejouer immédiatement.
- **Bonne Note** : Ajoute 100€ au solde du joueur.
- **Heure libre** : Avance le joueur de 2 cases.
- **Cours ennuyeux** : Le prochain lancé du joueur est divisé par 2.
- **Heure de colle** : Le joueur passe un tour.
- **Bureau du proviseur** : Le joueur perd 100€.
- **Contrôle surprise** : Le joueur perd 75€ s'il n'a pas révisé, sinon il ne perd rien.

Classe `Plateau`

Description

La classe `Plateau` représente le plateau du jeu Monopoly. Elle contient un tableau de cases (soit des terrains, soit des cases spéciales), permettant de gérer les déplacements et interactions des joueurs sur le plateau.

Constructeur

- `__init__(self)`

- **Effet** : Initialise le plateau avec un ensemble de cases. Certaines cases sont des terrains (instanciées à partir de la classe `Terrain`), et d'autres sont des cases spéciales (instanciées à partir de la classe `CaseSpeciale`).

Le tableau `cases` est une liste qui contient des objets représentant les différentes cases du plateau. Les cases sont disposées dans un ordre spécifique, et certaines sont de type "Chance", "Prison", "Réunion Parent-Prof", etc.

Méthodes

- `avoir_terrain(self, i)`
  - **Paramètre** :
    - `i` (int) : L'index de la case à récupérer.
  - **Retour** : Retourne le terrain ou la case à l'index spécifié dans la liste `cases`. Si l'index est invalide, retourne "Terrain inexistant".
  - **Exemple d'utilisation** :

```
plateau = Plateau()
terrain = plateau.avoir_terrain(3)  # Récupère la case à l'index 3 du plateau
print(terrain.nom)  # Affiche le nom du terrain à cet index
```

Attributs

- `cases` (list) : Une liste contenant les différentes cases du plateau. Les cases peuvent être soit des instances de la classe `Terrain`, soit des instances de la classe `CaseSpeciale`.

  Exemple d'initialisation :

```
self.cases = [
    CaseSpeciale("Départ", "départ"),
    Terrain("Sunny head beach", "T11"),
    Terrain("The Autistic Boulevard", "T11"),
    Terrain("Overthinking path", "T11"),
    Terrain("Hairy Bird Nest", "T24 BIS"),
    CaseSpeciale("Chance", "chance"),
    Terrain("Reptilian Way", "T22"),
    Terrain("Perinity City", "T22"),
    Terrain("The Market Bistro", "T22"),
    CaseSpeciale("Prison", "visite"),
    Terrain("The..hmmm...MeteoTower", "T21"),
    Terrain("Tagad island", "T21"),
    Terrain("The Musical Shopping-Center", "A104"),
    Terrain("Improvisation's City", "A104"),
    CaseSpeciale("Chance", "chance"),
    Terrain("The Useless Street", "T10"),
```

```
    Terrain("Juventus Cartel", "T10"),
    Terrain("Crazy City", "T10"),
    CaseSpeciale("Réunion Parent-Prof", "évenement"),
    Terrain("The Youth District", "T24"),
    Terrain("Sleepover'z Motel", "T24"),
    CaseSpeciale("En Prison", "en_prison"),
    Terrain("Happy gap-teeth city", "T24")
]
```