1 Classe Belligérant

Un belligérant capable de combattre à mains nues (et en pagne).

- Propriétés :
  1. nom : Le nom du belligérant.
     - Type : str
     - Accès : lecture seule
  2. pts_vies : Le nombre de points de vie courant du belligérant.
     - Type : int
     - Accès : accès complet
  3. pts_vies_max : Le nombre de points de vie initial du belligérant.
     - Type : int
     - Accès : lecture seule
  4. défense : Le facteur de défense du belligérant. Plus ce facteur est élevé, plus le belligérant pourra résister aux attaques.
     - Type : int
     - Accès : lecture seule
  5. force : Le facteur de force du belligérant. Plus ce facteur est élevé, plus les attaques du belligérant seront efficaces.
     - Type : int
     - Accès : lecture seule
  6. _items : Ensemble d'Items que possède le Belligérant.
     - Type : Ensemble d'Items
     - Accès : lecture seule
- Constructeur :
  1. __init__ : Initialise un belligérant. Outre le nom, fournit en paramètre, les valeurs de force, défense et de points de vie sont calculés de la façon suivante :
     - force : 1 dé12
     - défense : 1 dé12
     - pts_vie : 2 dé12 + 20
- Méthodes :
  1. nom : __str__ : Fournit une chaîne de caractère représentant le belligérant.

- Retour : Une chaîne représentant le belligérant de la forme «nom (F :*force* D :*défense* V :*pts_ vie*)
  - Type de retour : str

2. attaquer : Calcule le coefficient d'attaque d'un assaut par la formule.
   - Retour : Le coefficient d'attaque d'un assaut calculé. : *force* + 1 dé12
     - Type de retour : int
3. parer : Calcule le coefficient de parade lors d'un assaut.
   - Retour : Le coefficient de parade lors d'un assaut calculé par la formule : *défense* + 2 dé6
     - Type de retour : int
4. subir_dégâts : Après avoir reçu une attaque, cette méthode calcule les dégâts subis par le belligérant et les déduit de ses points de vie selon la formule : dégâts = impact.
   - Paramètres :
     (i) impact (int) : La force d'impact de l'attaque reçue.
   - Retour : Le nombre de points de vie réellement enlevé au Belligérant.
5. est_mort : détermine si un belligérant est mort.
   - Retour : Vrai si et seulement si pts_vie est 0 ou moins.
     - Type de retour : bool
6. prendre_item : Ajoute un item à ceux que possède le Belligérant. Le poids total de ses items ne peut excéder 5Kg par point de force.
   - Paramètres :
     (i) un_item (Item) : l'item à ajouter aux items du belligérant.
   - Assertions :
     (i) Le poids total des items (incluant celui ajouté) ne dépasse pas 5kg par point de force du Belligérant. Belligérants
       - Message : «L'item est trop lourd pour être ajouté au Belligérant».
7. jeter_item : Retire un item de la liste d'items détenus par le Belligérant.
   - Paramètres :
     (i) un_item (Item) : l'item à enlever.

8. Assertions :
   (i) L'item existe dans la liste.
      – Message : «L'item n'existe pas».

2 Classe Équipe

Regroupe les belligérants d'une même équipe (d'un même joueur).
– Propriétés :

1. nom : nom de l'équipe
   – Type : str
   – Accès : Lecture seule

– Constructeur :

1. __init__ : Initialise une équipe avec un nom mais sans belligérants immédiatement.
   – Paramètres :
      (i) nom (str) : Le nom de la nouvelle équipe.

– Méthodes :

1. __len__ : Retourne le nombre d'éléments dans l'équipe.
   – Retour : Le nombre de belligérants dans l'équipe
      – Type de retour : int
2. ajouter_belligérant : Ajoute un belligérant à l'équipe
   – Paramètres :
      (i) un_belligérant : le nouveau belligérant à ajouter à l'équipe.
         – Type : Belligérant
3. belligérant : Accesseur des belligérants
   – Paramètres :
      (i) indice : numéro du belligérant à retourner
         – Type : int
         – Valeur par défaut : None
      (ii) nom : nom du belligérant à retourner
         – Type : str
         – Valeur par défaut : None

- Retour : Si *indice* n'est pas None, retourne le belligérant numéro *indice* dans la liste, sinon, retourne le belligérant dont le nom correspond à *nom*. Si *indice* et *nom* sont None, retourne None.
  - Type de retour : Belligérant
- Assertions :
  (i) *indice* est None ou représente un élément existant de la liste de Belligérants
    - Message : «indice {*indice*} invalide»

  (ii) *nom* est None ou représente le nom d'un élément existant de la liste de Belligérants
    - Message : «nom '{*indice*}' est invalide»

4. est_éliminée : Retourne Vrai si l'équipe est éliminée.
   - Retour : Vrai si et seulement si tous les combattants de l'équipe sont morts.
   - Type de retour : bool

3 Classe Guerrier

Un belligérant de type Guerrier. Il peut porter une armure et manier une arme.

- Hérite de : Belligérant
- Propriétés :

  1. armure : L'armure que porte actuellement le guerrier
     - Type : Armure
     - Accès : complet
  2. arme : L'arme qu'utilise actuellement le guerrier. Le Guerrier ne peut utiliser une arme dont la classe dépasse la sienne.
     - Type : Arme
     - Accès : complet

- Constructeur :

  1. __init__ : Initialise un Guerrier, d'abord sans arme ni armure. Ses facteurs de force et de défense obtiennent un boni d'un dé12 par rapport au Belligérant et ses points de vie de 2xDé20.
     - Paramètres :

       (i) un_nom (str) : Le nom du nouveau Guerrier

- Méthodes :
  1. calculer_dégâts : Surdéfinition de la méthode Belligérant.calculer_dégâts(). En plus du calcul des dégâts comme pour n'importe quel belligérant, les dégâts réels sont réduits de le moitié de la classe de l'armure portée par le guerrier.
     - Paramètres :
       (i) impact (int) : La force d'impact de l'attaque reçue.
  2. Retour : Le nombre de points de vie qui doivent être retirés au Guerrier suite à l'attaque.
     - Type de retour : int
  3. attaquer : Surdéfinition de la méthode Belligérant.attaquer. Le coefficient d'attaque du belligérant est multiplié par la classe de l'arme qu'il utilise au moment de l'attaque.
     - Retour : Le coefficient d'attaque d'un assaut.
       - Type de retour : int
  4. arme : Mutateur de l'arme.
     - Paramètres :
       (i) une_arme (Arme) : L'arme qui doit être utilisée par le Guerrier. Le Guerrier ne peut utiliser une arme dont la classe dépasse la sienne.
     - Assertions :
       (i) La classe de l'arme n'est pas plus grand que celle du Guerrier. Belligérants
         - Message : «Le Guerrier ne peut utiliser une arme dont la classe est plus grande que la sienne.»
     - subir_dégâts : Surdéfinition de la méthode Belligérant.subir_dégâts(). Soustrait au Guerrier le nombre de points de vie calculé par calculer_dégâts. L'armure subit ensuite autant d'usure qu'un cinquième de l'impact.
       - Paramètres :
         (i) impact (int) : La force d'impact de l'attaque reçue.

4 Classe Mage

Un belligérant de type mage capable de jeter des sorts.

- Hérite de : Belligérant
- Propriétés :
  1. puissance : La puissance du mage. Pour lancer un sort, il ne peut jeter que des sorts dont la classe est inférieure ou égale à sa puissance.
     - Type : int
     - Accès : complet
  2. mana : La quantité d'«énergie» magique que possède le Mage. Il ne peut lancer de sort qui demande plus de mana que la quantité qu'il possède.
     - Type : int
     - Accès : complet
- Constructeur :
  1. __init__ : Initialise un Mage. Sa puissance est donnée par un dé6 et sa mana par 2xDé20. Initialement, il ne possède aucun sort.
     - Paramètres :
       (i) param (un_nom) : Le nom du Mage.
- Méthodes :
  1. jeter_sort : Jete un sort vers une cible. Pour réussir à lancer son sort, la différence entre la puissance du mage et la classe du sort doit être plus grande que le lancé d'un dé6. Dans tous les cas, la mana du mage est diminuée d'autant qu'il est requis par le sort.
     - Paramètres :
       (i) sort (Sort) : Le sort qui doit être jeté par le Mage
       (ii) cible (Belligérant) : Le belligérant vers qui le sort est jeté.
     - Assertions :
       (i) La puissance du Mage doit être au moins aussi grande que le classe de *sort*
           - Message : «Puissance (*puissance*) $<$ Sort.classe (*sorte.classe*)
       (ii) La mana du Mage doit être au moins aussi grande que la mana requise par le *sort*
           - Message : «Mana (*mana*) $<$ Sort.mana_requise (*sorte.mana_requise*)
  2. parer : Calcule le coefficient de parade lors d'un assaut. Puisque le Mage est plus agile que le Belligérant moyen, son coefficient la

valeur arrondie de 10% par point de puissance de plus que celui calculé par Belligérant.parer().
- Retour : Le coefficient de parade lors d'un assaut.
  - Type de retour : int

3. ajouter_sort : Ajoute un sort à la liste des sorts connus par le Mage.
   - Paramètres :
     (i) un_sort (sort) : Le nouveau Sort à ajouter au Mage.
   - Assertions :
     (i) Le mage ne doit pas déjà posséder ce sort
       - Message : «Le Sort (*un_sort*) existe déjà»

5 Classe Sort

Classe abstraite représentant un sort qui peut être jeté par un Mage.
- Propriétés :

1. mana_requise : quantité de mana requise pour jeter ce sort
   - Type : int
   - Accès : lecture seule
2. classe : la classe du sort, représente la difficulté à jeter ce sort.
   - Type : int
   - Accès : lecture seule

- Méthodes :

1. __str__ : Fournit une représentation en chaîne de caractères du Sort.
   - Retour : Une représentation du Sort sous la forme «classe *classe* ; mana requise : *mana_requise*»
2. activer : Méthode abstraite. Active le sort qui agit alors sur sa cible.
   - Paramètres :
     (i) cible (Belligérant) : Le belligérant ciblé par le sort.

6 Classe SortDartDeFeu

Sort qui lance un dart de feu vers un belligérant.
- Hérite de : Sort
- Constructeur :

  1. __init__ : Initialise le Sort avec sa classe (1) et sa mana (2).

- Méthodes :

  1. __str__ : Fournit une représentation en chaîne de caractères du Sort.
     - Retour : Une représentation du Sort sous la forme «Dart de feu : classe *classe* ; mana requise : *mana_requise*»
  2. activer : Active le sort qui agit alors sur sa cible. Il porte une attaque de puissance de 2xDé12. La cible peut parer le sort comme pour une attaque physique.
     - Paramètres :

       (i) cible (Belligérant) : Le belligérant ciblé par le sort.

7 Classe SortGuérison

Sort qui permet d'augmenter le nombre de points de vie d'un belligérant.
- Hérite de : Sort
- Constructeur :

  1. __init__ : Initialise le Sort avec sa classe (1) et sa mana (4).

- Méthodes :

  1. __str__ : Fournit une représentation en chaîne de caractères du Sort.
     - Retour : Une représentation du Sort sous la forme «Guérison : classe *classe* ; mana requise : *mana_requise*»
  2. activer : Active le sort qui agit alors sur sa cible. Il porte redonne 1 dé20 de points de vie à sa cible seulement si elle n'est pas morte, sinon, elle ne fait rien.
     - Paramètres :

       (i) cible (Belligérant) : Le belligérant ciblé par le sort.

8 Classe SortProtection

Sort qui augmente la protection d'un belligérant.
- Hérite de : Sort
- Constructeur :

  1. __init__ : Initialise le Sort avec sa classe (2) et sa mana (8).

- Méthodes :

  1. __str__ : Fournit une représentation en chaîne de caractères du Sort.
     - Retour : Une représentation du Sort sous la forme «Protection : classe *classe* ; mana requise : *mana_requise*»
  2. activer : Active le sort qui agit alors sur sa cible. Il augmente la défense du belligérant de 50%.
     - Paramètres :

       (i) cible (Belligérant) : Le belligérant ciblé par le sort.

9 Classe SortRésurrection

Description
- Hérite de : Sort
- Constructeur :

  1. __init__ : Initialise le Sort avec sa classe (4) et sa mana (15).

- Méthodes :

  1. __str__ : Fournit une représentation en chaîne de caractères du Sort.
     - Retour : Une représentation du Sort sous la forme «Résurrection : classe *classe* ; mana requise : *mana_requise*»
  2. activer : Active le sort qui agit alors sur sa cible. Il redonne à sa cible 2xDé20 points de vie.
     - Paramètres :

       (i) cible (Belligérant) : Le belligérant ciblé par le sort.

10 Classe Arme

Classe abstraite représentant une arme générique.
- Hérite de : Item
- Propriétés :
  1. classe : La classe de l'arme. Plus la classe est élevée, plus elle est destructrice.
     - Type : int
     - Accès : lecture seule
  2. bonus : Le bonus d'attaque accordé par l'arme. Cette propriété est calculée par les sous-classes.
     - Type : int
     - Accès : lecture seule
- Constructeur :
  1. __init__ : Initialise un Arme avec sa classe.
     - Paramètres :
       (i) une_classe (int) : La classe de l'arme.
     - Assertions :
       (i) La classe ne peut être négative
           - Message : «La classe (*une_classe*) est invalide»
- Méthodes :
  1. __str__ : Méthode abstraite qui retourne une représentation en chaîne de caractère de l'arme sous la forme «Type d'arme (classe)».
     - Retour : une représentation en chaîne de caractère de l'arme sous la forme «Type d'arme (classe)».
       - Type de retour : str
  2. bonus : Méthode abstraite qui calcule le bonus accordé par cette arme pour une attaque.
     - Retour : Le bonus accordé par cette arme.
       - Type de retour : int

11 Classe Armure

Classe abstraite représentant une armure générique.

- Hérite de : Item
- Propriétés :
  1. classe : La classe de l'armure. Plus la classe est élevée, plus elle protège celui qui la porte.
     - Type : int
     - Accès : lecture seule
  2. bonus : Le bonus de défense accordé par l'arme. Cette propriété est calculée par les sous-classes.
     - Type : int
     - Accès : lecture seule
  3. usure : Le pourcentage d'usure de l'armure. Initialement à 0, elle augent progressivement jusqu'à 100 ; l'armure est alors rendue inutile.
     - Type : int
     - Accès : complet
- Constructeur :
  1. __init__ : Initialise un Armure avec sa classe.
     - Paramètres :
       (i) une_classe (int) : La classe de l'armure.
     - Assertions :
       (i) La classe ne peut être négative
           - Message : «La classe (*une_classe*) est invalide»
- Méthodes :
  1. __str__ : Méthode abstraite qui retourne une représentation en chaîne de caractère de l'armure. sous la forme «Type d'armure (classe)».
     - Retour : une représentation en chaîne de caractère de l'armure sous la forme «Type d'armure (classe)».
       - Type de retour : str
  2. bonus : Méthode abstraite qui calcule le bonus accordé par cette armure pour une attaque.
     - Retour : Le bonus accordé par cette armure.
       - Type de retour : int

12 Classe Dé

Simule un dé avec un nombre variable de faces.

- Méthodes :

    1. lancer : Méthode de classe qui simule un lancer de dé.
        - Paramètres :
            (i) faces (int) : le nombre de faces du dé à lancer.
                - Valeur par défaut : 6
        - Retour : Un nombre au hasard entre 1 et *faces* inclusivement.
            - Type de retour : int
        - Assertions :
            (i) *faces* doit être > 1
                - Message : «Le nombre de faces doit être > 1»

13 Classe Item

Classe abstraite représentant un item générique pouvant être acquis et porté par un Belligérant.

- Propriétés :

    1. nom : poids
        - Type : int
        - Accès : lecture seule

- Constructeur :

    1. __init__ : Initialise l'item avec son poids.
        - Paramètres :
            (i) un_poids (int) : le poids de l'item en grammes.

- Méthodes :

    1. __str__ : Donne une représentation en chaîne de caractères de l'item sous la forme «p=*poids*».
        - Retour : La représentation en chaîne de caractères de l'item.
            - Type de retour : str

14 Classe Potion

Classe abstraite représentant une potion buvable par un Belligérant.

- Hérite de : Item
- Propriétés :
  1. parts : Le nombre de parts que contient la fiole de potions.
     - Type : int
     - Accès : lecture seule
- Constructeur :
  1. __init__ : Initialise la potion avec son nombre de parts.
     - Paramètres :
       (i) nb_part (int) : Le nombre de parts contenues dans la fiole de potion.
     - Assertions :
       (i) Le nombre de parts ne peut être < 1.
           - Message : «nb_parts ne peut être < 1.»
- Méthodes :
  1. faire_boire : Fait boire la potion par un belligérant. Décrémente de un le nombre de parts restantes. L'effet de la potion est exécutée par la méthode *activer* des sous-classes.
     - Paramètres :
       (i) cible (Belligérant) : Le Belligérant qui doit boire la potion.
     - Assertions :
       (i) Le nombre de parts ne peut être < 1.
           - Message : «nb_parts ne peut être < 1.»
  2. activer : méthode abstraite qui exécute l'effet de la potion sur la cible
     - Paramètres :
       (i) cible (Belligérant) : Le Belligérant qui doit boire la potion.
     - Assertions :
       (i) Le nombre de parts ne peut être < 1.
           - Message : «nb_parts ne peut être < 1.»

15 Classe PotionGuérison

Description
- Hérite de : Potion
- Propriétés :
    1. pts_de_vie_par_part : Le nombre de pts de vie redonnés par cette potion pour chaque part bue.
        - Type : int
        - Accès : lecture seule
- Constructeur :
    1. __init__ : Initialise la PotionGuérison avec son nombre de parts et de points de vie par part.
        - Paramètres :
            (i) nb_part (int) : Le nombre de parts contenues dans la fiole de potion.
            (ii) pts_de_vie_par_part : Le nombre de pts de vie redonnés par cette potion pour chaque part bue.
        - Assertions :
            (i) Le nombre de parts ne peut être < 1.
                - Message : «nb_parts ne peut être < 1»

16 Classe Joueur

Un joueur qui contrôle une équipe de Belligérants.
- Attributs :
- contrôle : Un Contrôle par lequel le joueur peut interagir avec le jeu.
    - Type : Contrôle
- Propriétés :
    1. nom : Le peudonyme du joueur
        - Type : str
        - Accès : lecture seule
    2. équipe : L'équipe que contrôle le joueur
        - Type : Équipe
        - Accès : lecture seule

3. actions_par_tour : Le nombre d'actions que peut faire le joueur à chaque tour.
   - Type : int
   - Accès : complet

- Constructeur :

1. __init__ : Initialise le Joueur avec un nom sans espaces au début et à la fin et un Contrôle.
   - Paramètres :
     (i) un_nom (str) : Le pseudonyme du Joueur
     (ii) un_contrôle (Contrôle) : Le Contrôle par lequel le joueur interagira avec le jeu.
   - Assertions :
     (i) Le nom ne peut être une chaîne vide.
       - Message : «nom invalide»

- Méthodes :

1. __str__ : Donne une représentation en chaîne de caractères du Joueur.
   - Retour : Le nom du joueur.
     - Type de retour : str
2. créer_belligérant : Crée un belligérant et l'ajoute à l'équipe du joueur. Utilise le contrôle pour permettre au joueur de choisir le type et le nom de son belligérant.
3. jouer : utilise le contrôle pour permettre au joueur de jouer son tour. Un tour comporte les étapes suivantes :
   (i) Choisir un belligérant dans son équipe.
   (ii) Choisir une action que ce belligérant doit effectuer.
   (iii) Si l'action est une attaque, choisir son adversaire.
   (iv) Si l'action concerne un item, choisir l'item
   (v) Si l'action est jeter un sort, choisir le sort et la cible
   (vi) Effectuer l'action sur la cible.

   Ces étapes sont répétées tant que le nombre d'actions par tour du joueur n'est pas atteint ou qu'il n'y pas choisit de passer son tour.

4. choisir_belligérant : Utilise le contrôle pour permettre au joueur de choisir un belligérant dans sa propre équipe.
   - Retour : Le belligérant sélectionné.
     - Type de retour : Belligérant
5. choisir_action : Utilise le contrôle pour permettre au joueur de choisir sa prochaine action.
   (i) Paramètres :
      i. un_acteur (Belligérant) : Le belligérant qui doit faire l'action. La liste des actions proposées sera construite en fonction du type et des capacités de ce Belligérant.
      ii. Retour : L'action choisie par le joueur
         - Type de retour : Action
6. choisir_cible : Utilise le contrôle pour permettre au joueur de choisir un belligérant ciblé par une action. Le belligérant peut être de son équipe ou de celle d'un adversaire.
   - Paramètres :
     (i) les_équipes (Liste d'Équipes) : La liste de toutes les équipes parmi lesquelles le joueur peut choisir sa cible.
   - Retour : Le belligérant sélectionné.
     - Type de retour : Belligérant

17 Classe Contrôle

Classe abstraite représentant un Contrôle capable de faire interagir le joueur avec le jeu. Le Contrôle permet au joueur de faire sélectionner plusieurs types d'éléments. Des méthodes abstraites permettent de communiquer directement avec le joueur (par l'écran et le clavier, par exemple).

- Méthodes :

1. choisir_Objet
   - Paramètres :
     (i) choix (Liste d'Objets) : Liste des objets parmi lesquels choisir. (Par exemple, une liste de Belligérants ou d'Items.)
   - Retour : L'objet sélectionné.
     - Type de retour : Objet

18 Classe ContrôleXXX

Classe représentant un Contrôle capable de faire interagir le joueur avec le jeu. Le Contrôle permet au joueur de faire sélectionner plusieurs types d'éléments. Des méthodes abstraites permettent de communiquer directement avec le joueur (par l'écran et le clavier, par exemple).

- Hérite de : Contrôle
- Méthodes :
  1. choisir
     - Paramètres :
       (i) choix (Liste de str) : Liste des choix offerts au joueurs.
     - Retour : Le numéro du choix sélectionné.
       - Type de retour : int
  2. afficher_message : Permet d'afficher un message au joueur sans attendre de réponse de sa part.
     - Paramètres :
       (i) un_message (str) : Le message à afficher.
       (ii) confirmation (bool) : détermine si la méthode doit attendre que le joueur ait confirmé la lecture du message pour continuer.
         - Valeur par défaut : False
  3. saisir : Permet au joueur de saisir un texte.
     - Paramètres :
       (i) question (str) : La question posée au joueur. Aucune question si None.
         - Valeur par défaut : None
     - Retour : Le texte saisi par le joueur.
       - Type de retour : str.

19 Classe Partie

Une partie de Tournoi Pythonesque 4 entre deux ou plusieurs joueurs.

- Attributs :
  1. _joueurs : Les joueurs de cette partie.
     - Type : Liste de joueurs

2. _tour : Le numéro du tour courant. 0 pendant la préparation du jeu
   - Type : int
3. _items_épars : Les items laissés sur le champ de bataille. Lorsqu'un belligérant meurt, ses items sont remis ici.
   - Type : Liste d'Items.
4. Constructeur :

   (i) __init__ : Initialise la partie.
   - Paramètres :
     i. les_joueurs (Liste de Joueurs) : Liste de tous les participants à la partie.
   - Assertions :
     i. La liste doit contenir au moins deux joueurs
        - Message : «Il n'y a pas suffisamment de joueurs.»

5. Méthodes :

   (i) populer_équipes : Permet de créer tous les belligérants de toutes les équipes. Tour à tour, chaque joueur est invité à créer son belligérant. Le joueur qui commence est choisi au hasard.

   (ii) créer_belligérant : Permet de créer un belligérant. Selon le mode de partie choisie, les belligérants sont créés ici au hasard ou en demandant au joueur de choisir certaines caractéristiques (par sa méthode créer_belligérant).

   (iii) Paramètres :
     i. un_joueur (Joueur) : Le joueur pour lequel créer le belligérant. Si None, le belligérant est créé au hasard.

   (iv) Retour : Le belligérant créé.

   (v) démarrer_partie : Joue la partie jusqu'à ce que toutes les équipes sauf une soient éliminées. Le premier joueur à jouer est choisi au hasard. À tour de rôle, les joueurs jouent leur tour. Dès qu'une équipe est éliminée, le joueur devient spectateur.

   (vi) jouer_tour : Joue un tour de jeu. Tous les joueurs, à tour de rôle jouent leur tour.

(vii) répartir_items : Après la création des belligérants, mais avant le début de la partie, les items épars sont répartis entre les belligérants. Le joueur qui commence est choisi au hasard puis tous, à tour de rôle sélectionnent un item et un belligérant pour le prendre. Jusqu'à ce qu'il n'y ait plus d'items ou que tous les belligérants aient atteint la limite de ce qu'ils peuvent porter.