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 du belligérant.
   — Type : int
   — Accès : accès complet
3. défense : Le facteur de défense du belligérant. Plus ce facteur est élevé, plus le belligérant pourra résister aux attaques.
   — Type : int
   — Accès : lecture seule
4. force : Le facteur de force du belligérant. Plus ce facteur est élevé, plus les attaques du belligérant seront efficaces.
   — Type : int
   — 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.
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

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»

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
     - 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. 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*)
      — La man du Mage doit être au moins aussi grande que la mana requise par le *sort*
         — Message : «Mana (*mana*) < Sort.mana_requise (*sorte.mana_requise*)
   — parer : Calcule le coefficient de parade lors d'un assaut. Puisque le Mage est plus agile que le Belligérant moyen, son coefficient est 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
2. 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. 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 et sa mana.
— Paramètres :

(i) une_classe (int) : La classe du sort (1)

(ii) une_mana_requise (int) : La quantité de mana requise par le sort (2)

— Méthodes :

1. 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 et sa mana.
   — Paramètres :
   (i) une_classe (int) : La classe du sort (1)
   (ii) une_mana_requise (int) : La quantité de mana requise par le sort (4)

— Méthodes :

1. activer : Active le sort qui agit alors sur sa cible. Il porte redonne 1 dé20 de points de vie à sa cible.
   — 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 et sa mana.
   — Paramètres :
   (i) une_classe (int) : La classe du sort (2)
   (ii) une_mana_requise (int) : La quantité de mana requise par le sort (8)

— Méthodes :

1. 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 et sa mana.
   — Paramètres :
   (i) une_classe (int) : La classe du sort (4)
   (ii) une_mana_requise (int) : La quantité de mana requise par le sort (15)

— Méthodes :

1. 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.
— 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.

- 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

Description

- 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»