**Projet POO - RPG**

https://github.com/Nykio/ProjetPOO.git

Thème :Moyen-Age / Médiéval-Fantaisie

| **Class** | **État** |
| --- | --- |
| Cartes | **Approuvé** |
| Combats | **Approuvé** |
| Joueur | **Approuvé** |
| Potions | **Approuvé** |
| Attaque | **Approuvé** |

**Utilisation :**

Cartes :

* Faire des combats
* Points de vies
* Attaques
* Statistique de vitesse ( déterminant qui attaque en premier )

Combats :

* Utilisation d’objets
* Prends deux cartes
* Première carte qui n’a plus de PV perds le combat et envoie la prochaine jusqu'à ce que le joueur n’a plus de cartes

Dresseurs ( ennemis ) :

* Leurs propres cartes

Joueur ( contient un deck pour ranger des cartes et un inventaire avec des objets ):

**Organisation du code :**

* Class Carte
  + Attributs :

PV

Vitesse

Attaques

Element

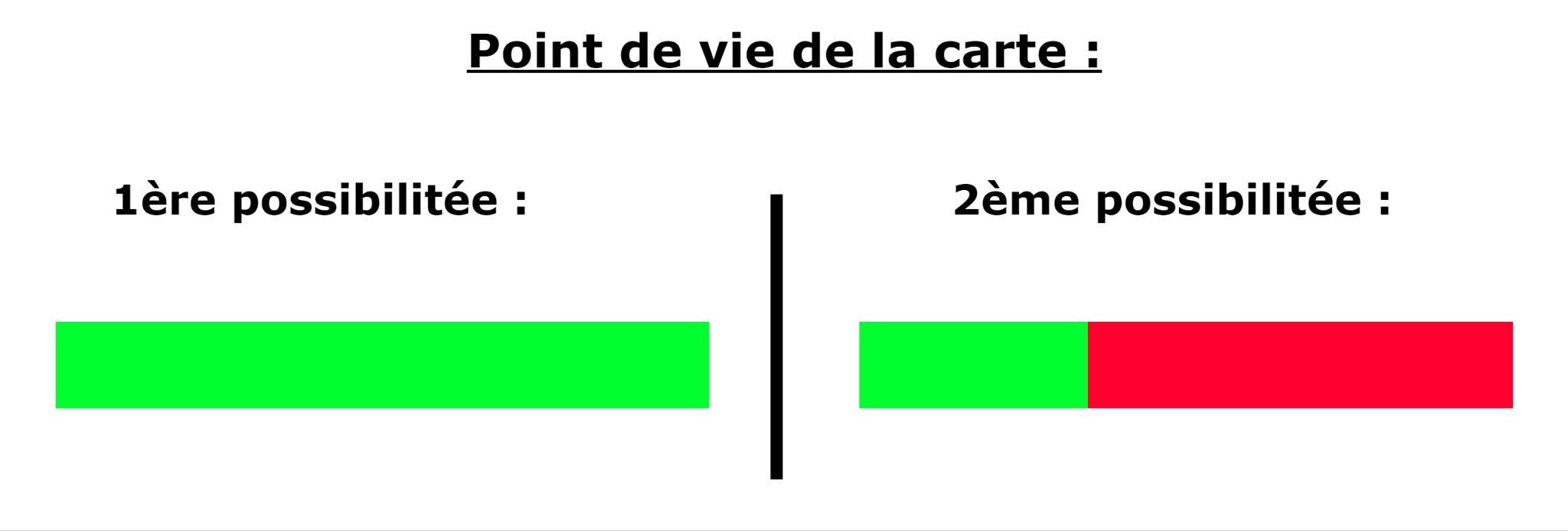
* Méthodes :

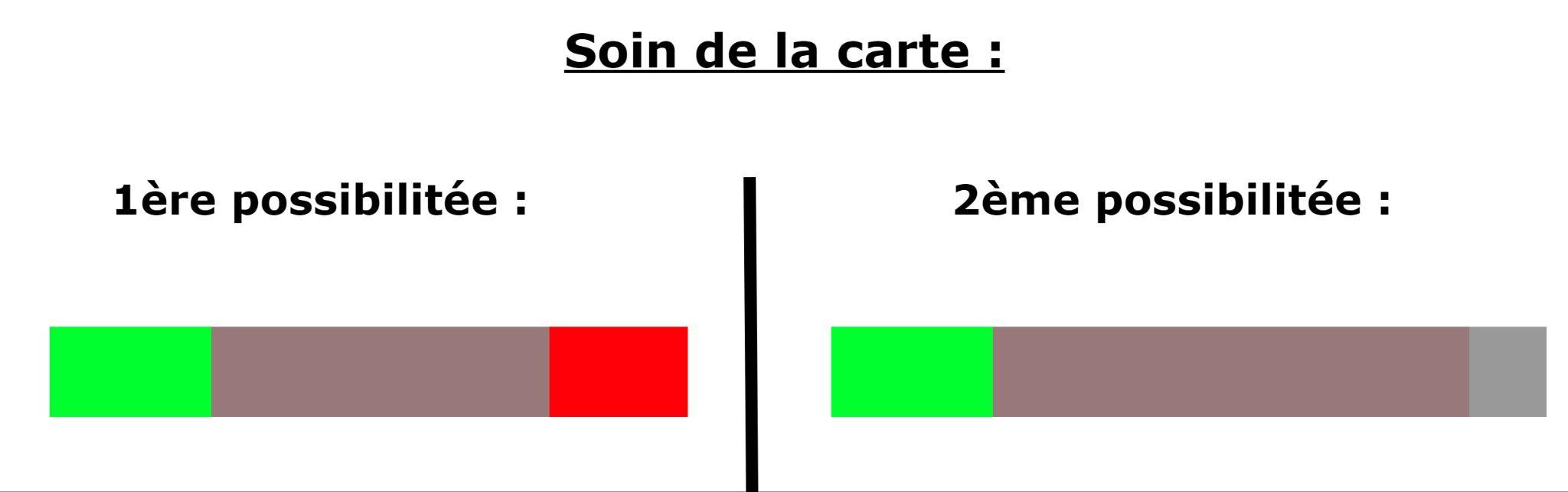
Apprendre attaque :

Fait apprendre une attaque à la carte

Soigner :

Applique une potion à la carte , lui faisant regagner les PV perdus





Choix auto :

Une méthode choix auto qui permet de choisir une attaque aléatoirement ( peut utiliser **random** )

Est en vie :

Nous disant si oui ou non la carte a encore des pv

Choix Attaque :

Pour choisir une attaque

* Class Joueur
  + Attributs :

Deck ( qui contient les cartes )

Sac ( Inventaire du joueur )

* Méthodes :

Ajouter au sac

Sortir carte :

Sors une carte du deck pour l’utiliser en combat

Ajouter carte :

Ajoute une carte

Choisir carte :

Une méthode qui permet de choisir une carte parmi celle de son deck ( elle ne doit pas retirer la carte du deck )

* Class Attaques
  + Attributs :

Dégâts

Nom

* Class Combats
  + Attributs :

Un joueur ennemi

* Méthodes :

Lancer combat

Utiliser objet

Un combat fonctionnerait avec un système de tour par tour , chaque tour le joueur peut soit soigner sa carte , lancer une attaque ou changer la carte sur le terrain .

Les méthodes attendus sont :

* Class Potions
  + Attributs :

Soin

**Organisation du travail :**

A faire :

Code :

* Class Cartes :
  + choix automatique d’une attaque
  + ajouter des types
* Class Attaques
  + ajouter des types
* Class Joueur
* Class Combats
* Class Soin

Ajouter des types : faiblesses , résistance

Ajouter des statuts liés à des effets : sommeil , paralysie etc …

Readme

Specs

| Chose à faire dans les vacances : | | | |
| --- | --- | --- | --- |
| Aucun type Tâche | Aucun type Description | Personne Propriétaire | Menus déroulants État |
| Equilibre des types | Les cartes et attaques disposent d’une nature qui ont un avantage et un inconvénient dans la table des types | Toine | Approuvée |
| Readme |  | Toine | Approuvée |
| Specs |  | Toine | Approuvée |
| choix\_auto ( dans la class Carte ) | Une méthode qui choisit une attaque aléatoirement parmi celle de la carte , indice : from random import | Maïna | Approuvée |
| est\_en\_vie ( dans la class Carte ) | Une méthode qui revient True si les points de vie de la carte sont strictement supérieur à 0 , autrement False | Maïna | Approuvée |
| choix\_attaque | Une méthode qui retourne une attaque parmi celle des cartes que le joueur choisit | Toine | Approuvée |
| choisir\_carte ( dans la class Joueur ) | Une méthode qui choisit  une carte pour l’utiliser en combat parmi celle du deck/équipe | Mélaine | Approuvée |
| choisir\_carte\_aleatoire ( dans la class Joueur ) | Une méthode qui choisit  une carte aléatoirement pour l’utiliser en combat parmi celle du deck/équipe | Mélaine | Approuvée |
| afficher\_sac ( dans la class Joueur ) | Une méthode qui permet de visualiser les objets de notre inventaire | Mélaine | Approuvée |
| afficher\_deck ( dans la class Joueur ) | Une méthode qui permet de visualiser les cartes de notre deck | Mélaine | Approuvée |
| Class Combat |  | Toine | Approuvée |

Toine :

Class Attaque

Class Potion

Class Combat

#sur github



Maïna :

Class Carte

#sur github



Mélaine :

Class Joueurs

#sur github

def affiche\_sac(self):

if self.sac:

print("Contenue du sac :")

for objet in self.sac:

print(objet.nom)

else:

print("Le sac est vide :")

def affiche\_deck(self):

if self.deck:

print("Contenue du deck :")

for objet in self.deck:

print(objet.nom)

else:

print("Le deck est vide :")



**Introduction des Types :**

