Rapport du Jeu de carte la bataille Python

MARCH 2

Réalisé par: AJJA Anass & TAHIRI Reda

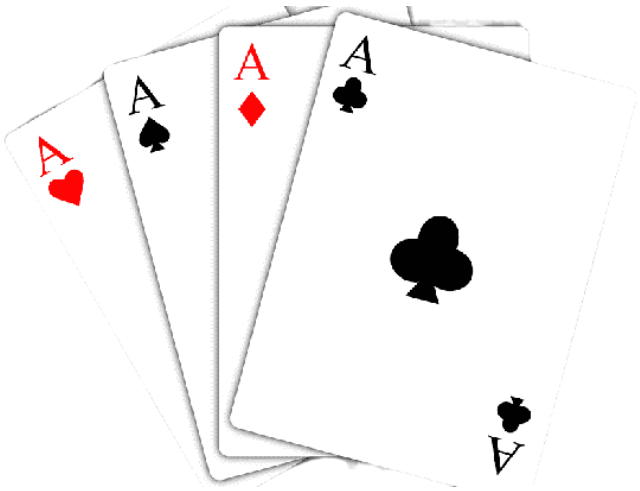
MAROC ynov CAMPUS

Sommaire:

I. Introduction

Présentation générale du projet Jeu de carte la Bataille et son objectif :

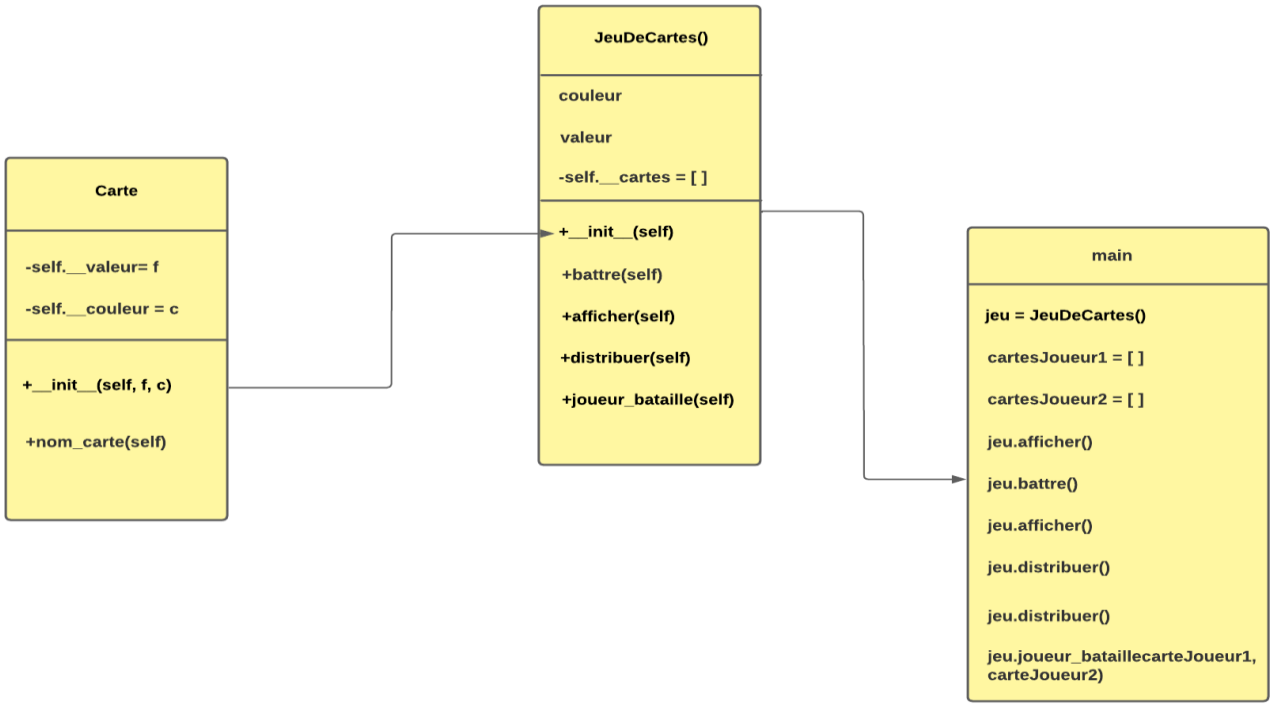
"Bataille" est un jeu de cartes simple et amusant qui peut être joué par deux joueurs ou plus. Le but du jeu est de gagner toutes les cartes en jeu en les obtenant des autres joueurs. Chaque joueur reçoit un nombre égal de cartes, puis ils retournent simultanément la carte du dessus de leur pile. Le joueur avec la carte la plus élevée remporte toutes les cartes en jeu. En cas d'égalité, les joueurs continuent à retourner une carte chacun jusqu'à ce qu'un joueur ait une carte plus élevée que l'autre. Le jeu se poursuit jusqu'à ce qu'un joueur ait toutes les cartes ou qu'un nombre déterminé de manches ait été joué. Bataille est un jeu de hasard simple, mais qui peut être excitant et amusant pour tous les âges.



II. Gestion de Projet

Dans ma gestion de projet, j'ai utilisé les outils Trello et Discord pour m'assurer que tout se déroule de manière organisée et efficace. J'ai utilisé Trello pour créer des tableaux de suivi de projet, y compris des listes de tâches à effectuer, des tâches en cours et des tâches terminées. Cela m'a permis de visualiser rapidement l'avancement du projet et de m'assurer que toutes les tâches sont terminées à temps. J'ai également utilisé Discord pour communiquer avec mon coéquipier , en créant un canal dédié au projet. Cela nous a permis de rester connectés et de nous tenir informés des derniers développements en temps réel, ce qui a été essentiel pour la réussite du projet. En utilisant Trello et Discord de manière cohérente, j'ai pu maintenir une bonne organisation et une communication fluide tout au long du projet.

III. Diagramme UML



Pour créer un UML, nous avons commencé par déterminer les différents objets et leurs relations les uns aux autres dans le système que nous souhaitions modéliser. Ensuite, nous avons choisi les diagrammes UML appropriés pour représenter ces objets et relations, tels que les diagrammes de classes, de séquence, de cas d'utilisation, entre autres. Nous avons alors utilisé un outil de modélisation UML pour dessiner les diagrammes en respectant les conventions et les symboles standards de l'UML. Une fois les diagrammes terminés, nous les avons examinés pour vérifier leur précision et leur cohérence avec les spécifications du système. Enfin, nous les avons utilisés pour communiquer les concepts et les détails du système aux autres membres de l'équipe de développement.

IV. Programmation Python

1. Programme Jeu de carte la bataille :

class: Carte

On a créé une classe appelée "Carte". La classe Carte représente une carte à jouer.

def__init__ (self, f, c):

La méthode __init__ a deux attributs privés (__valeur et __couleur) qui sont initialisés avec les paramètres f et c.

def nom_carte (self) :

La méthode nom_carte retourne une chaîne de caractères représentant le nom de la carte. Cette méthode concatène les attributs __valeur et __couleur de la carte pour former la chaîne de caractères qui est ensuite retournée.

class : Jeu de Carte

On a créé une classe Jeu de carte qui représente un jeu de cartes.

def __init__(self):

La méthode __init__, définit deux listes : couleur et valeur, qui contiennent respectivement les noms des couleurs des cartes (Pique, Coeur, Carreau, Trefle) et les valeurs des cartes (As, 2, 3, 4, 5, 6, 7, 8).

Ensuite, à l'aide de deux boucles imbriquées, on crée une liste de 52 cartes, qui contient toutes les combinaisons possibles de valeurs et de couleurs. Pour chaque couleur dans la liste couleur, on itère sur chaque valeur dans la liste valeur, et pour chaque combinaison de valeur et de couleur, on crée une nouvelle instance de la classe Carte avec cette valeur et cette couleur, et on l'ajoute à la liste self.__cartes.

def battre (self) :

La méthode battre utilise la fonction shuffle de la bibliothèque random pour mélanger la liste des cartes (self.__cartes) de manière aléatoire. shuffle est une fonction qui prend une liste en entrée et modifie cette liste en place pour la mélanger.

def afficher (self) :

La méthode def afficher utilise une boucle for pour parcourir chaque carte dans la liste self.__cartes, et pour chaque carte, elle appelle la méthode nom_carte de l'objet Carte pour obtenir son nom complet sous forme de chaîne de caractères.

def distribuer(self) :

La méthode def distribuer vérifie si la liste des cartes dans le jeu (self.__cartes) est vide. Si c'est le cas, elle affiche un message indiquant qu'il n'y a plus de cartes et retourne None.

Sinon, la méthode utilise la fonction randint de la bibliothèque random pour générer un nombre aléatoire entre 0 et le nombre de cartes restantes dans le jeu. Ce nombre aléatoire t est utilisé pour sélectionner une carte aléatoire dans la liste self.__cartes.

La carte sélectionnée est ensuite supprimée de la liste self.__cartes à l'aide de la fonction del ou pop pour éviter qu'elle ne soit distribuée à nouveau.

def jouer_bataille(self, joueur1, joueur2) :

La méthode jouer_bataille prend en paramètres deux listes de cartes joueur1 et joueur2 représentant les mains des deux joueurs respectifs.
On a utilisé la boucle while qui continue tant que les deux joueurs ont encore des cartes dans leurs mains.
Les deux joueurs tirent chacun une carte de leur jeu avec la méthode pop() qui retire la dernière carte de la liste et la renvoie. Les cartes tirées sont stockées dans les variables carte_joueur1 et carte_joueur2.
Les noms des cartes jouées par chaque joueur sont affichés avec print().
Si les cartes ont la même valeur, une "bataille" est déclenchée : les cartes jouées sont stockées dans la liste cartes_bataille et les joueurs tirent chacun une carte face cachée. Si ces deux cartes ont également la même valeur, la bataille continue jusqu'à ce que des cartes distinctes soient tirées. Le joueur ayant tiré la carte de la plus haute valeur remporte la bataille et les cartes jouées sont ajoutées à son jeu avec la méthode extend(). Si les deux joueurs ont épuisé leur jeu avant qu'une carte de plus haute valeur ne soit tirée, la bataille s'arrête.
Si les cartes ont des valeurs différentes, le joueur ayant tiré la carte de la plus haute valeur remporte la manche et les cartes jouées sont ajoutées à son jeu avec la méthode extend().
Après chaque manche ou bataille, le nombre de cartes restantes dans le jeu de chaque joueur est affiché.
Une fois que l'un des joueurs a épuisé toutes ses cartes, la boucle while s'arrête et un message indiquant le vainqueur du jeu est affiché.

main

Tout d'abord, un objet JeuDeCartes est créé et affiché avec la méthode afficher().
Ensuite, le jeu est mélangé avec la méthode battre() et réaffiché avec afficher().

Ensuite, 16 cartes sont distribuées à chaque joueur à l'aide de la méthode distribuer() du jeu, et les cartes sont stockées dans les listes cartesJoueur1 et cartesJoueur2. Les mains de chaque joueur sont affichées.

Enfin, la méthode jouer_bataille() est appelée avec les listes de cartes des deux joueurs. Cette méthode simule un jeu de bataille jusqu'à ce qu'un joueur n'ait plus de cartes. Les cartes sont tirées de chaque main à tour de rôle et comparées pour déterminer le vainqueur de chaque manche. Si les deux cartes ont la même valeur, une bataille est déclenchée, pendant laquelle chaque joueur tire une carte face cachée et une carte face visible. Si un joueur remporte la bataille, il récupère toutes les cartes en jeu. Le gagnant est ensuite déterminé en fonction du nombre de cartes que chaque joueur a en sa possession. Tout au long du jeu, des messages sont affichés pour informer les joueurs de ce qui se passe.

Enfin, une fois que l'un des joueurs n'a plus de cartes, le jeu est terminé et le gagnant est affiché.

2. Programme Jeu de carte la bataille 52 cartes :

On a juste remplacer les 32 par 52 cartes en ajoutant les valeurs 9, 10, Dame, Valet, Roi pour la méthode __init__ dans la classe Jeu de Carte, ainsi on a remplacé le nombre 16 par 26 dans la méthode main enfin de nous afficher les 52 cartes et distribuer 26 cartes pour chaque joueur .

3. Programme Jeu de carte la bataille non découverte :

Premièrement on a utilisé le premier programme jeu de carte la bataille de 32 cartes, mais afin d’avoir un jeu de carte la bataille non découverte, on a pu changer dans la méthode jouer_bataille en class Jeu de Carte en ajoutant une boucle en répétant le tirage du carte par chaque joueur deux fois en cas de bataille.

V. Conclusion

Synthèse de résultats et des leçons apprises

La réalisation de ce projet nous a permis de mettre en pratique les compétences en développement que nous avons acquises au cours de notre formation. Nous avons appris à utiliser les outils et les technologies tels que Python, Pycharm et Trello pour gérer efficacement le développement de notre projet de Jeu de carte bataille.
En ce qui concerne les résultats obtenus, nous avons réussi à implémenter toutes les fonctionnalités prévues dans le temps imparti de 7 jours. Les tests unitaires effectués ont montré que notre jeu fonctionne correctement et répond aux exigences fonctionnelles définies.
Cependant, nous avons également rencontré des défis tout au long du projet, tels que la gestion efficace des tâches et la résolution de bugs. Pour y faire face, nous avons utilisé Trello pour diviser les tâches et suivre l'avancement du projet, et nous avons travaillé en équipe pour résoudre les problèmes rencontrés.
En conclusion, ce projet nous a permis de développer nos compétences en développement de logiciels et de mieux comprendre les défis liés à la gestion de projet. Les leçons apprises au cours de ce projet nous seront très utiles pour les projets futurs et nous aideront à devenir des développeurs plus expérimentés et compétents.

VI. Remerciements

Tout d'abord, nous tenons à remercier les membres de notre groupe de travail pour leur engagement et leur dévouement envers ce projet. Chacun a apporté ses compétences et son savoir-faire pour créer un produit final de qualité.
Nous souhaitons également remercier notre encadrant pour son soutien et ses précieux conseils tout au long de ce projet. Sa guidance a été précieuse pour nous aider à surmonter les défis rencontrés et à atteindre nos objectifs.
Encore une fois, merci à tous pour votre soutien et votre aide tout au long de ce projet. Nous espérons que vous avez apprécié simuler notre code "Jeu de carte la Bataille" autant que nous avons apprécié le développer !