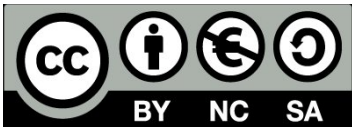
TP POO, piles et file

Antoine MAROT (antoine.marot@ensemblescolaire-niort.com)

David SALLÉ (david.salle@ensemblescolaire-niort.com)

Julien SIMONNEAU (julien.simonneau@ensemblescolaire-niort.com)



Version du document v0.1

Date 06/07/2020

Table des matières

1 - Quelques exemples

1.1 - Programmation orientée objet

1.1.1 - Compte bancaire

- Créer une classe nommée CompteBancaire qui représente un compte bancaire, ayant pour attributs : numero_compte (type numérique), nom (nom du propriétaire du compte de type chaîne), solde.
- Créer un constructeur ayant comme paramètres : numéro_compte, nom, solde.
- Créer une méthode versement() qui gère les versements sur le compte.
- Créer une méthode retrait() qui gère les retraits.
- Créer une méthode agios() permettant d'appliquer les agios à un pourcentage de 5% du solde.
- Créer une méthode afficher() permettant d'afficher les détails sur le compte.

1.1.2 - Tournoi (optionnel)

L'objectif de cet exercice est de créer des personnages dont vous choisirez le nom, les caractéristiques (force, constitution et agilité) et l'équipement (arme et armure) afin de combattre en duel.

A la création du personnage, vous aurez 60pts à répartir entre les caractéristiques et le coût de l'arme et armure.

Les points de vie, l'initiative, la chance de toucher l'adversaire et les dégâts au combat sont déterminés par exemple de la manière suivante :

- PV = constitution * 10
- base_initiative = agilité + init(arme) + init(armure)
- attaque = force + agilité
- %toucher = attaque(attaquant) - défense(défenseur) + 40
- dégâts totaux = force + dégâts(arme)

Vous pourrez imaginer des armes et armures. Par exemple :

- bâton (coût : 5, dégâts : 1-5, init : 20)
- épée (coût : 12, dégâts : 5-10, init : 10)
- hache à 2 mains (coût : 20, dégâts : 10-20, init : 0)
- aucune armure (coût : 0, défense : 0, init : 20)
- armure légère (coût : 10, défense : 10, init : 10)
- armure lourde (coût : 20, défense : 20, init : 0)

Le combat se déroule en tours successifs jusqu'à ce qu'un joueur perde tous ses points de vie. A chaque tour, l'initiative de chaque joueur est augmentée

de la base_initiative. Celui qui possède la plus grande initiative attaque, mais son initiative perdra alors 40 points pour le tour suivant.

- Créer des classes Arme et Armure ainsi que les méthodes cout(), degats(), defense(), initiative()
- Créer une classe Personnage ayant pour attributs : nom (type chaîne), force, constitution, agilité (de type entier) ainsi qu'arme et armure (de type chaîne).
- Créer les méthodes points_vie(), attaque(), total_degats() et base_initiative().
- Ecrire la fonction attaquant(joueur1, joueur2) qui renvoie le joueur qui possède la plus grande initiative
- Ecrire la fonction toucher(attaquant, defenseur) qui renvoie un booléen
- Ecrire la fonction combat(joueur1, joueur2) qui renvoie le nom du vainqueur
- Compléter le programme avec la création des personnages puis le combat qui en découle

1.2 - Piles et files

1.2.1 - Panier

Adam aime offrir des pommes. Il décide d'aller en cueillir suffisamment pour remplir son panier, qui peut en contenir 25. Au rythme d'une pomme cueillie toutes les 10 secondes, il pourrait remplir son panier rapidement s'il n'était pas tenter d'en croquer une toutes les trois pommes cueillies. Guère raisonnable, il succombe à la tentation deux fois sur 3 en moyenne.

En utilisant le panier d'Adam comme instanciation de la classe Pile, modéliser la situation en donnant la durée de remplissage.

1.2.2 - File d'attente

Lors de la pause de 10h01, les toilettes de la cafet' sont prises d'assaut : toutes les 20s, il y a 2 chances sur 5 qu'une personne rejoigne la file d'attente. En revanche, toutes les minutes, les toilettes se libèrent pour que le premier de la file se soulage.

Modéliser la file d'attente à l'aide de la classe File que vous avez créée.

Donner la taille de la file d'attente à 10h21.

En estimant qu'une personne arrivant dans la file d'attente a une petite envie (niveau 1) et que cette envie augmente chaque minute (d'un niveau), quel est le niveau d'envie de la personne en tête de file à 10h21 ?

1.2.3 - Tours de Hanoï

Les tours de Hanoï sont un jeu de réflexion consistant à déplacer des disques de diamètres différents d'une tour de départ à une tour d'arrivée en passant par une tour intermédiaire en un minimum de coups, tout en respectant les règles suivantes :

- on ne peut déplacer plus d'un disque à la fois,
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

Modéliser les tours de Hanoï à l'aide de la classe Pile.

2 - Projet solitaire

2.1 - Présentation du projet

Dans cette partie, vous allez devoir réaliser un jeu de solitaire à partir d'un jeu de cartes en utilisant les classes Carte, JeuCarte, Pile et File que vous avez créées.

2.2 - Principe du jeu

Ce jeu peut être mis en œuvre avec un jeu de 32 cartes ou un jeu de 52 cartes.

Au départ, les cartes sont mélangées et regroupées dans un tas appelé talon.

À partir de ce talon, on remplit 4 piles de défausse avec des cartes masquées qui contiennent respectivement 4, 3, 2 et 1 cartes. Seules les cartes situées au dessus de chaque pile de défausse sont visibles.

Le but du jeu est de ranger toutes les cartes par famille dans 4 emplacements spécifiques, un par famille. Pour cela, on peut réaliser à chaque tour une des actions suivantes :

- retourner une carte du talon qui devient visible et qui prend la place de la carte précédemment retournée,
- ranger la carte située sur le talon dans un des emplacements famille, une nouvelle carte du talon est alors automatiquement retournée,

- ranger une des cartes situées sur les zones de défausse dans un des emplacements famille
- défausser la carte située sur le talon dans une des zones de défausse,
- défausser une des cartes situées sur les zones de défausse dans une autre zone de défausse,
- défausser une des cartes situées dans un emplacement famille vers une des zones de défausse.

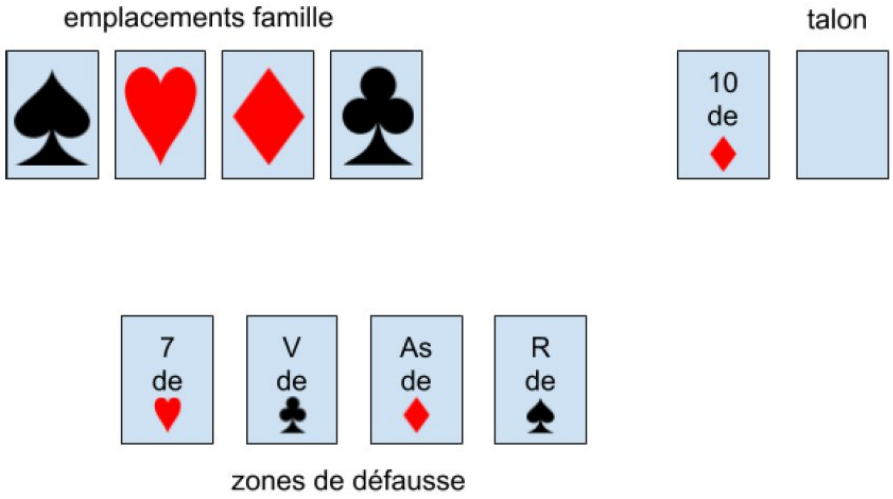
Lors de la réalisation de ces actions, Il faut respecter deux règles :

- pour ranger les cartes dans les emplacements famille, les cartes doivent être positionnées de la plus petite à la plus haute.
- pour défausser une carte dans une zone de défausse, la carte doit être d'une couleur différente que celle qui la précède et doit avoir un rang immédiatement inférieur.

Lorsque le talon est vide, on reprend les cartes retournées non encore positionnées et on redémarre avec un nouveau talon qui redonne d'abord la carte retournée non positionnée la plus ancienne.

2.3 - Schéma

Voici un schéma permettant de visualiser les différents éléments du jeu :



On considère que le jeu est proposé avec un paquet de 32 cartes.

À ce moment là du jeu, aucune carte n'a été positionnée dans un des emplacements famille, la talon a commencé à être retourné. La carte visible est le 10 de carreau.

On peut au choix, déplacer le 10 de carreau sur la zone de défausse avec le valet de trèfle, ou bien ranger le 7 de cœur dans l'emplacement famille des cœurs, ou encore déplacer le roi de pique sur l'As de carreau.

2.4 - Mise en œuvre

2.1.1 - Partie obligatoire

Vous devez créer une interface dans la console Python permettant de visualiser à chaque étape l'état des différents éléments du jeu (emplacements famille, piles de défausse et talon) et offrant la possibilité de jouer en saisissant une touche au clavier parmi :

- t : on retourne une carte du talon
- d : on effectue un déplacement qui peut-être une défausse ou un rangement
- x : on abandonne la partie en cours.

Les éléments utilisés seront issus des classes Carte, JeuCarte, Pile et File. Vous devrez remettre une archive contenant les différentes classes utilisées et le fichier jeu en tant que tel.

Au départ, vous devrez offrir la possibilité de choisir entre un jeu de 32 cartes ou de 52 cartes.

Vous devrez enfin rechercher un moyen de savoir si la partie est achevée ou non et dans le cas où elle est est achevée, vous devrez informer le joueur du nombre de fois où il a retourné le talon.

2.4.2 Partie optionnelle

Voici différents éléments que vous pouvez ajouter :

- imaginer un score qui évolue en fonction des coups réalisés,
- créer une interface graphique associée au jeu,
- créer une variante où les cartes sont tirées 3 par 3 du talon avec visibilité des 3 cartes mais accès possible uniquement à la carte située sur le dessus. Si celle-ci est déplacée, on peut alors avoir accès à celle du dessous, etc...
- rendre visibles les cartes positionnées par le joueur sur une zone de défausse,