UE Programmation Orientée Objet

TD Pierre-Papier-Ciseaux

Le jeu est probablement bien connu. Il se joue à deux joueurs. Lors d'un tour de jeu, les joueurs jouent simultanément[1] et disposent de 3 coups possibles : *pierre*, *papier*, *ciseaux*.
La résolution d'un tour de jeu est la suivante :

1. si les deux joueurs jouent la même chose, le coup est nul,
2. la *pierre* bat (casse) les *ciseaux*,
3. le *papier* bat (enferme) la *pierre*,
4. les *ciseaux* battent (coupent) le *papier*.

Un coup nul rapporte 1 point, un coup victorieux 2 points et un coup perdant 0.
Une partie se joue en une suite de tours de jeu en nombre fixé au départ. Le vainqueur est celui (quand il y en a un) qui a le plus de points à l'issue de tous les tours, sinon la partie est nulle.

Nous allons procéder à l'analyse d'un programme objet permettant de jouer à ce jeu. Il est important de traiter les questions dans l'ordre. Exceptionnellement, il est même préférable de ne pas lire une question avant d'avoir répondu aux précédentes.

Q 1 . Donnez l'algorithme d'un traitement qui permet de faire jouer un tour de jeu (méthode `playOneRound` d'une classe `Game`).

Q 2 . En partant de la réponse à la question précédente, déterminez et définissez les types nécessaires à la modélisation de ce jeu.

Ecrivez le code java de la méthode `playOneRound` de `Game`.

NB : si cela vous aide, pour cette question vous pouvez considérer que les joueurs jouent en choisissant aléatoirement l'un des trois coups possibles.

Q 3 . On souhaite maintenant que l'un des joueurs (ou les deux) ne joue plus aléatoirement mais applique une autre stratégie de jeu, comme celle de jouer toujours le coup *papier* par exemple.

Que proposze-vous de modifier à ce qui a été défini précèdemment pour prendre en compte cette modification ?

Q 4 . Pour traiter la question précedente, êtes-vous en mesure de faire une proposition qui permet de ne pas modifier la classe `Game`, et en particulier la méthodes `playOneRound`, définies dans les deux premières questions ?

Si oui, laquelle ?

Q 5 . On souhaite en fait pouvoir varier autant que possible les stratégies appliquées par chacun des deux joueurs. En plus des deux stratégies déjà évoquées on peut ainsi en imaginer d'autres telles que :

- ▷ toujours jouer *pierre*,
- ▷ toujours jouer *feuille*,
- ▷ jouer en boucle *pierre* suivi de *papier*, ou toute autre séquence,
- ▷ etc.

A nouveau :

Que faut-il modifier à ce qui a été défini précèdemment pour prendre en compte toutes ces différentes stratégies ?

Faites une proposition. Est-il nécessaire de modifier la classe `Game` ?

Q 6 . Enfin, comme dernière stratégie de jeu on ajoute la possibilité pour un joueur humain de choisir le coup à jouer par une saisie au clavier.

Que faut-il faire pour intégrer cette stratégie de jeu ?

Faites une proposition.

Aléatoire voir la classe `java.util.Random`

Saisie au clavier voir la classe `util.Input` fournie dans le semainier pour le TP

[1] D'un point de vue du programme cela signifie que lorsqu'un joueur choisit son coup de jeu, il n'a aucune information sur ce que choisit l'autre joueur au même tour.