TP 2 : Course de dé

Pour ce TP nous nous proposons de programmer un petit jeu de dé ayant une conception objet simple et dont les règles sont exposées ci-dessous.

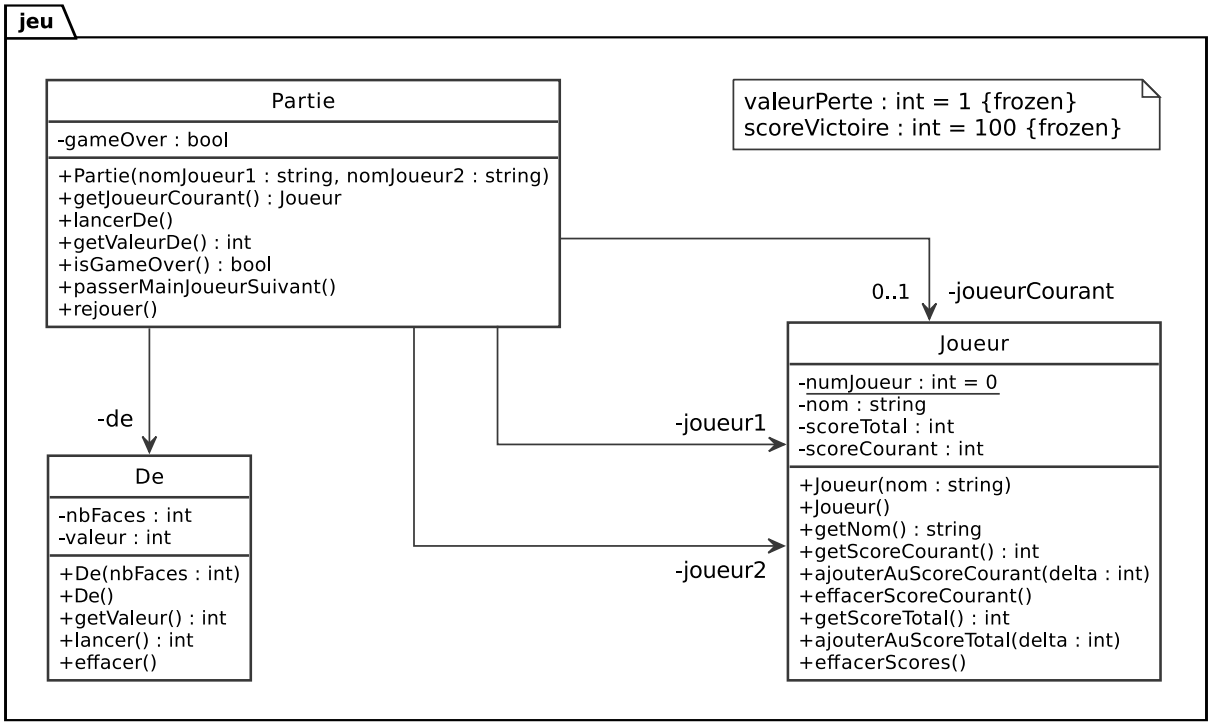
1 Règles du jeu

La « course de dé » se joue en lançant un dé. Les joueurs jouent tour à tour (pour ce TP nous nous limiterons à deux joueurs). Lorsque c'est à son tour, le joueur lance un dé et, tant qu'il ne fait pas 1, accumule les points des lancers successifs. S'il fait 1, il perd les points qu'il avait accumulés pendant son tour de jeu et la main passe, c'est à l'autre joueur de jouer. Un joueur peut sciemment décider d'arrêter de lancer le dé et ainsi sécuriser le score de son tour de jeu. Le premier joueur qui atteint un score de victoire (par exemple 100) gagne.

2 Conception à implémenter

Le diagramme de classes suivant vous présente la conception à coder, mais il n'explicite pas si les attributs sont des pointeurs, des références ou aucun des deux. À vous de réfléchir et de faire les choix judicieux. On ne veut pas de copies d'objets superflues si possible.

La contrainte `{frozen}` permet d'indiquer ici une constante connue à la compilation (mot clé `constexpr` en `C++`). La note n'est reliée à rien pour signifier ici que ces constantes seront directement dans le namespace `jeu`.



2.1 Le dé

La classe `De` représente un dé à `nbFaces` faces (comme ça vous pourrez la réutiliser dans les jeux de rôles que vous créerez. Un dé 20… pas d'problème ;-))

- Un dé peut être créé en spécifiant son nombre de faces; sans cela il aura par défaut 6 faces
- On peut lancer un dé et ceci mettra à jour la valeur du dé avec un nombre aléatoire compris dans l'intervalle [1, `nbFaces`] (cf. doc de la fonction `rand()`, ou bien de la classe `uniform_int_distribution` pour les plus ambitieux); cette valeur sera aussi retournée par la méthode
- On pourra à tout moment consulter la dernière valeur du dé grâce à la méthode `getValeur()`; cette méthode renverra 0 si le dé n'a jamais été lancé (nous verrons plus tard comment gérer ces cas de figure autrement, grâce aux exceptions)
- Effacer le dé signifie remettre le dé dans son état initial comme s'il n'avait jamais été lancé

Pensez à régulièrement compiler et tester chaque fonctionnalité du dé.

2.2 Un joueur

La classe `Joueur` permet de conserver toutes les données d'un joueur lors d'une partie de course de dé. Les méthodes portent un nom suffisamment explicite pour que vous puissiez deviner ce qu'elles font.

- Précisons juste qu'un joueur peut être créé à partir de son nom, ou bien, si celui-ci n'est pas précisé, les joueurs s'appelleront "Joueur 1", "Joueur 2" et ainsi de suite (oui, peut-être que bientôt vous aurez l'envie de faire évoluer le jeu pour pouvoir jouer à autant de joueurs que vous voulez)
- L'opérateur d'insertion (<<) sera surchargé pour formater un joueur dans un flux de la façon suivante :

  "`nom` (`scoreTotal`) a pour l'instant cumulé `scoreCourant`"

  (cf. exemple d'un déroulement de partie en fin de sujet)

Pensez à régulièrement compiler et tester chaque fonctionnalité du joueur.

2.3 Une partie

La classe `Partie` va contenir les informations permettant de gérer un duel entre deux joueurs.
La logique d'une partie est principalement regroupée au sein des deux méthodes `lancerDe()` et `passerMainJoueurSuivant()`.

- `lancerDe()` va effectuer un lancer du dé et, suivant le résultat, soit augmenter le score courant du joueur de la valeur du dé, soit effacer le score courant du joueur et passer la main à l'autre joueur (attention, on ne peut lancer un dé que si la partie n'est pas terminée)
- `passerMainJoueurSuivant()` va terminer le tour du joueur courant en mettant ses scores à jour et en vérifiant si la partie doit continuer ou non; si la partie est terminée l'attribut associé est mis à jour, sinon on change de joueur courant
- on doit pouvoir faire une revanche, puis une belle si nécessaire entre les deux mêmes joueurs; c'est le rôle de la méthode `rejouer()` qui réinitialise la partie, le joueur perdant étant le premier à jouer la nouvelle partie

Les autres méthodes sont juste des getters classiques.

Pensez à régulièrement compiler et tester chaque fonctionnalité de la partie.

2.4 La course de dé

Le programme principal, i.e. votre fonction `main()` (qui ne devra pas être dans le namespace `jeu`), consistera simplement à créer une partie et à jouer de façon à respecter les règles du jeux. Vous afficherez le déroulement de la parie sur la sortie standard. Vous pouvez vous inspirer du déroulement de la page suivante pour coder votre jeu.

Vous pourrez aussi rajouter, s'il vous reste du temps, un prompt proposant de faire une revanche ou une belle suivant la situation (on peut le faire dans le `main()`, mais il faudra probablement faire évoluer la conception pour ajouter une classe et les attributs nécessaires si vous voulez le faire proprement en version Objet).

Notez que vous aurez besoin de faire des entrées clavier pour savoir ce que les joueurs désirent faire comme action. Pour cela vous pourrez utiliser l'opérateur d'extraction (>>) sur l'objet `cin` dont je vous invite à consulter la documentation si vous ne connaissez pas… faites vous plaisir.

Je ne sais pas si je l'ai déjà dit, mais pensez à régulièrement compiler et tester votre code ;-)

Voici un exemple de déroulement d'une partie en 30 points :

```
Carine et Laurent sont sur le point de commencer une partie endiablée de course de dé !
Carine (0) a pour l'instant cumulé 0
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Carine lance le dé et fait un 3
Carine (0) a pour l'instant cumulé 3
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : l
Carine lance le dé et fait un 3
Carine (0) a pour l'instant cumulé 6
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Carine lance le dé et fait un 3
Carine (0) a pour l'instant cumulé 9
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Carine lance le dé et fait un 4
Carine (0) a pour l'instant cumulé 13
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : S
Carine décide de sécuriser son score et totalise 13
Laurent (0) a pour l'instant cumulé 0
Laurent appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Laurent lance le dé et fait un 1
La main passe, dommage.
Carine (13) a pour l'instant cumulé 0
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : l
Carine lance le dé et fait un 6
Carine (13) a pour l'instant cumulé 6
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : l
Carine lance le dé et fait un 3
Carine (13) a pour l'instant cumulé 9
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : s
Carine décide de sécuriser son score et totalise 22
Laurent (0) a pour l'instant cumulé 0
Laurent appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Laurent lance le dé et fait un 1
La main passe, dommage.
Carine (22) a pour l'instant cumulé 0
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Carine lance le dé et fait un 6
Carine (22) a pour l'instant cumulé 6
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : L
Carine lance le dé et fait un 3
Carine (22) a pour l'instant cumulé 9
Carine appuyez sur L pour lancer le dé ou S pour sécuriser votre score : S
Carine décide de sécuriser son score et totalise 31
Carine gagne la partie !
```