Travail 1

Remise : mardi 11 février avant minuit
via l'option remise de Léa
enlevez les 2 dossiers debug (ou x64) et le dossier caché .vs
Pondération : 8%

But :

Les buts visés par ce travail sont de :

- Mettre en pratique les notions de base concernant la programmation objet
- Utiliser des relations d'association entre les classes
- Se familiariser à la gestion de versions avec Git

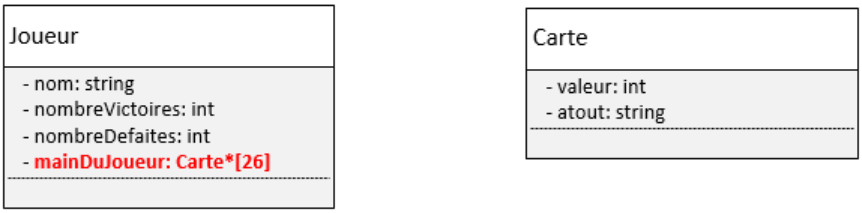
Ce premier travail doit s'effectuer de façon individuelle.

Description :

Le travail consiste à programmer un petit jeu de cartes en mode console. Les règles du jeu sont expliquées plus bas et sont très simples. Le schéma qui suit sera utilisé. Il n'y a pas de classe PaquetCarte afin de rendre le programme le plus simple possible. La relation d'association est utilisée pour indiquer les cartes que chaque joueur possède.

Schéma de classes

Schéma à implanter



Directives générales

- Chaque classe doit être programmée à l'intérieur de ses propres fichiers (.h et .cpp).
- Respectez le standard de programmation (doc 0.1 sur Léa).
- Respectez les bonnes techniques de programmation (doc 0.1).

- Faites attention pour que le nom des méthodes commence par un verbe.
- Faites attention pour donner les bonnes responsabilités aux classes.
- Respectez les principes de la programmation objet.
- Autant que possible, utilisez vos objets.
- Utilisez les constantes indiquant le nombre d'éléments maximum dans les tableaux.
- Sans trop vous attarder à créer un bel affichage (on est juste en mode console), faites-en sorte que l'on repère facilement les différentes valeurs affichées.

Programmation des classes

Git : Ouvrez le projet fourni et démarrez le suivi. Faites un 1er commit incluant les fichiers **nécessaires** seulement.

- La classe Carte est fournie, utilisez-la telle quelle.
- Programmez la classe Joueur en respectant le schéma d'implantation.
  - Notez que des pointeurs sont utilisés pour garder les cartes d'un joueur. Ainsi on utilise les cartes du 'paquet' et on crée une relation avec ces cartes.
  - Faites le constructeur par défaut seulement
  - Le Set() est permis pour le nom du joueur seulement. Pour les victoires et les défaites, il faut prévoir une autre façon de procéder pour augmenter les victoires et les défaites (respect du Tell don't ask).
  - Prévoyez une méthode qui permettra d'ajouter une carte à la main d'un joueur (un pointeur vers la bonne carte). Les cartes s'ajoutent une à la suite de l'autre dans le tableau représentant la main.
  - Il faut aussi une méthode pour enlever toutes les cartes de la main d'un joueur. Attention, on ne veut pas détruire la carte mais seulement indiquer que le joueur n'a plus de cartes (pas de delete).
  - Il est possible d'ajouter d'autres méthodes.

Git : Effectuez un nouveau commit significatif et envoyez le tout sur Github.

Programmation de la classe donnees

- Elle doit contenir le tableau de 52 cartes (représente le paquet) ainsi que 2 joueurs : à déclarer de façon 'public'.
- Faites une méthode permettant de créer les 52 cartes du tableau avec les bonnes valeurs. Cette méthode doit être appelée par le constructeur, ainsi les cartes seront automatiquement initialisées.
- Vous pouvez ajouter d'autres méthodes si vous le désirez mais ce n'est pas obligatoire.

Git : Effectuez un nouveau commit et envoyez le tout sur Github.

Programme principal et déroulement du jeu

Note : Faites des commits aussitôt qu'une partie est fonctionnelle. N'attendez pas que tout le programme soit terminé. Chaque commit doit être accompagné d'un commentaire significatif.

- Un objet 'leJeu' (type Donnees) est déclaré globalement. Ainsi, le constructeur est exécuté et les cartes sont créées. Utilisez cet objet.
- La fonction main() est fournie et permet de jouer plusieurs tours.
- Une fonction InitialiserJoueurs() existe. Programmez cette fonction afin de demander et d'assigner des noms aux 2 joueurs.
- Une fonction Jouer() existe. Utilisez cette fonction pour programmer un tour du jeu :
  - Il faut demander à l'usager le nombre de cartes à distribuer (max 26 cartes).
  - Les cartes du paquet doivent être mélangées et il faut donner le bon nombre de cartes à chaque joueur.
  - Pour chaque joueur, il faut afficher son nom, ses cartes (valeur et atout) et afficher le total de ses cartes.
  - Le joueur dont le total est le plus élevé gagne le tour et obtient ainsi une victoire tandis que l'autre joueur obtient une défaite.
  - Le nom du joueur qui obtient la victoire doit s'afficher.
- Le programme principal permet de jouer plusieurs tours. Lorsque l'usager choisit de terminer, il faut afficher, pour chaque joueur : son nom, son nombre de victoires et son nombre de défaites. Finalement, il faut indiquer le nom du grand gagnant.

Git : Effectuez un dernier commit et envoyez le tout sur Github.

Critères de correction et répartition des points :

Critères

- Respect du schéma
- Application des règles de la programmation objet
- Respect des exigences demandées
- Respect du standard de programmation et des bonnes pratiques de programmation
- Utilisation adéquate des objets
- Fonctionnement
- Interface

Répartition des points

- Déclaration et manipulation des données des joueurs **25 pts**
- Déclaration et manipulation des cartes des joueurs **25 pts**
- Déclaration et manipulation des cartes du paquet **25 pts**
- Déroulement du jeu **25 pts**