**Programmation Objet 2**

**Travail 2**

**Pondération : 10%**

**Remise : vendredi 2 novembre avant minuit**

**Note : Enlevez les dossiers bin et obj avant de remettre votre travail.**

Le but de ce travail est de mettre en pratique la notion d’interface tout en continuant à appliquer une structure MVC ainsi que des principes de programmation de ‘clean coding’ à l’aide du langage C#.

Le travail peut se faire de façon individuelle ou en équipe de 2. Vous devez m’indiquer les noms des coéquipiers aujourd’hui.

**Exigences**

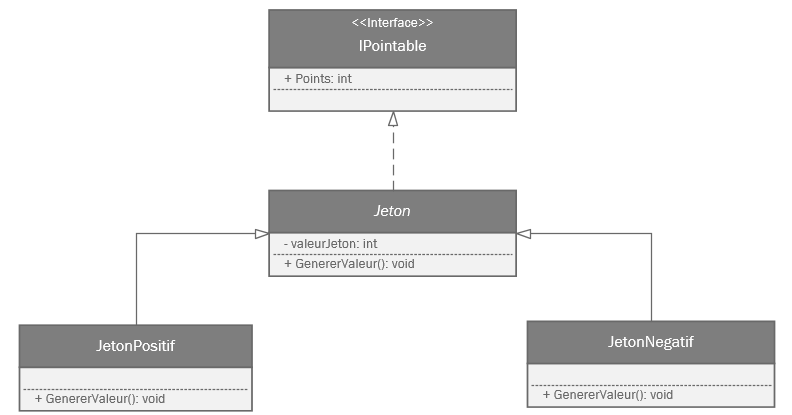
* Vous utilisez l’application fournie.
* Vous devez porter une attention particulière aux éléments qui suivent. Il est à noter que  
   **50% de la note du travail est dédié au respect de ces éléments**
  + architecture MVC
  + responsabilités des classes et principe « Tell don’t ask ».
  + notion de propriété
  + déclenchement et gestion des exceptions
  + respect du standard de programmation tel que présenté au cours (sur Léa)
  + découpage en fonctions et noms utilisés

**Description du travail, partie 1**

Le travail consiste à programmer un petit jeu où les joueurs peuvent acquérir des objets attribuant des points. Le but est évidemment d’obtenir le plus de points possibles grâce aux objets. Il existe différents types d’objets (des dés, des cartes et des jetons). Chaque type d’objet génère ses points de façon différente.

Étant donné que les objets à acquérir n’appartiennent pas à une même hiérarchie alors nous utiliserons une interface afin de donner un comportement commun à l’ensemble de nos classes. Les classes devront implémenter cette interface. Il sera donc possible de faire du **polymorphisme d’interface** dans le travail.

* Utilisez l’interface IPointable qui est fournie
* Ajoutez la classe De du cours 1 dans la solution
  + Modifiez son namespace afin d’utiliser celui de la solution
  + Faites-en sorte que la classe implémente l’interface : les points doivent correspondre à la valeur du dé.
* Ajoutez la classe carte du travail1
  + Modifiez son namespace
  + Faites-en sorte qu’elle implémente l’interface : les points doivent correspondre à la valeur de la carte.
  + Modifiez aussi le constructeur afin qu’il génère une carte.
* Pour les jetons, créez les classes respectant la hiérarchie du schéma qui suit



* + La classe jeton doit être abstraite
  + Les valeurs générées par les classes :
    - Doivent utiliser une formule math (pas de if)
    - jetonPositif : la valeur générée doit être entre 5 et 50
    - jetonNegatif : la valeur générée doit être -2 ou -4
  + Les points doivent correspondre à la valeur du jeton.
  + La valeur du jeton doit être automatiquement générée lors de la construction d’un nouveau jeton.
* Une classe joueur est fournie car le jeu peut se jouer à plusieurs joueurs. Présentement, cette classe contient seulement le nom du joueur et ses points accumulés. Vous devez compléter cette classe.
* Le contrôleur contient :
  + Une enum indiquant les types d’objets possibles. Utilisez cette enum afin de rendre le programme plus significatif.
  + Un tableau de string aussi est déclaré, lequel contient les chaines de caractères qui correspondent aux valeurs de l’enum. Ce tableau peut être utilisé pour faciliter l’affichage.
  + Une liste de joueurs.
  + Un attribut ‘tour’ indiquant quel joueur doit jouer.
* Comme d’habitude, la Form possède un objet du type du contrôleur. Le visuel du jeu est déjà fait mais vous pouvez le modifier si vous voulez.

**Fonctionnalités à programmer**

* Au début de l’exécution :
  + 2 joueurs doivent être créés
  + des noms doivent leur être assignés
  + les noms doivent s’afficher à l’écran
  + le nom du joueur courant doit s’afficher
* Le bouton ‘Obtenir un objet’ doit :
  + générer un type d’objets au hasard : dé, carte, jeton positif ou jeton négatif
  + faire afficher le type d’objet obtenu (vous ne devez pas utiliser GetType() mais les valeurs de l’enum) et sa valeur
  + les points de l’objet doivent être ajoutée aux points du joueur courant
  + les points du joueur doivent s’afficher
* Un clic sur le bouton ‘Joueur suivant’ doit :
  + permettre de passer au joueur suivant
  + mettre l’affichage à jour

**Description du travail, partie 2**

* Modifiez la classe joueur afin de garder la liste de tous les objets obtenus par le joueur.
* Lorsqu’un nouveau type d’objet est généré, ajoutez cet objet dans la liste des objets du joueur courant.
* Dans la Form, ajoutez un bouton et un listBox. Le bouton doit permettre de faire afficher dans le listBox tous les objets obtenus (nom du type d’objet et la valeur de ses points) par le joueur courant.

**Note** : Vous pourriez créer une interface INommable pour faire afficher le type d’objet. Je donne **10 points bonus** à ceux qui le font avec une possibilité d’avoir 110/100 sur le travail ☺ !

**Critères de correction**

* Respect des principes de la POO
* Respect des standards de programmation
* structure MVC et principe « Tell don’t ask »
* Fonctionnement
* Respect des exigences