TP 5

Les objectifs de ce cinquième TP sont :

- l'utilisation de LINQ
- à travers LINQ : l'utilisation de méthodes d'extension, d'expressions lambda et de types anonymes
- l'utilisation du pattern standard des événements

Le TP est pour cela composé de deux exercices distincts.

Exercice 1 : LINQ

Dans une solution, ajoutez le projet giRandoCore fourni avec l'énoncé. Créez une application Console dont giRandoCore sera une référence.

À l'aide d'instructions LINQ :

1) affichez tous les sommets de plus de 1000 mètres d'altitude
2) affichez le nom, le dénivelé et la distance de toutes les randonnées dont le dénivelé est supérieur à 300m et la distance supérieure à 7km
3) affichez le nom et la distance de toutes les randonnées de la plus courte à la plus longue
4) affichez les pays et le nombre de randonnées des pays ayant le plus de randonnées
5) affichez le nom et la vitesse moyenne des randonnées par ordre de vitesse moyenne croissante
6) modifiez la classe `giRandoCore.Guide` en lui ajoutant une propriété permettant de ne rendre que les randonnées à pied
7) groupez les randonnées par pays, puis pour chaque groupe, affichez le nom du pays et ses randonnées (nom, dénivelé et distance)
8) transformez les randonnées en un dictionnaire dont la clé sera le pays contenant les randonnées et la valeur sera la moyenne des vitesses moyennes, puis affichez toutes les paires clé-valeur.

Exercice 2 : événements
(examen du 2 novembre 2010)
Sujet - Jeu du morpion

OBJECTIFS

L'objectif de cet examen est de réaliser une application Console qui permet de regarder deux joueurs automatiques jouer au Morpion. Pour vous aider, deux bibliothèques vous sont fournies :

• la première `giMorpionCore.dll` contient des classes et des méthodes gérant une grande partie du jeu (démarrage du jeu, insertion de pièces, indication au prochain joueur qu'il doit jouer, test de la fin du jeu...),

• la deuxième `giMorpionTools.dll` contient une classe et une méthode statiques permettant d'afficher la grille du Morpion dans une fenêtre Console.

Mais alors que reste-t-il alors à faire ?

Ces bibliothèques ne communiquent pas, et il n'y a pas d'application. Dans la classe `Game` :

• la méthode `NotifyNextPlayer` appelle la méthode `Play` du prochain joueur (qui joue ainsi son coup),

• la méthode `Start` appelle la méthode `NotifyNextPlayer` sur le premier joueur,

• la méthode `InsertPiece` se contente d'insérer ou non la pièce et d'indiquer qui sera le prochain joueur, mais la liaison avec le reste du jeu n'est pas réalisée,

• la méthode `IsGameOver` teste si le jeu est terminé mais n'est jamais appelée.

Dans la classe `Player`, l'appel de la méthode `Play` appelle la méthode `InsertPiece` de `Game` (notez que cette méthode `InsertPiece` est virtuelle !!)

Vous avez la charge de :

• réutiliser cette bibliothèque, sans avoir accès au code source,

• écrire la liaison entre toutes ces méthodes pour que le jeu puisse se dérouler,

• créer des événements pour pouvoir renseigner une application extérieure sur le déroulement du jeu,

• créer une application Console permettant de visualiser le jeu et son déroulement en s'abonnant à ces événements.

QUESTION 1 : BIBLIOTHÈQUE ET ÉVÉNEMENTS

Vous devez réaliser une bibliothèque de classes `giMorpionExam.dll` contenant vos nouvelles classes pour le déroulement du jeu du Morpion, c'est-à-dire une classe `MyGame` dérivant de la classe `Game`, dans laquelle vous devez :

- ajouter des événements (et donc peut-être des classes :-) permettant de renseigner le reste du monde que :
  - le jeu a commencé,
  - le jeu est terminé (en renseignant qui a gagné, et quelle est la ligne gagnante),
  - un joueur vient d'être renseigné qu'il doit jouer (et qui est ce joueur),
  - une pièce a été insérée (où et par qui).

  Pour cela, vous pourrez choisir de réécrire des méthodes virtuelles, ou d'écrire de nouvelles méthodes. Dans tous les cas, vous devrez profiter de l'héritage de `Game` et éviter de réécrire des choses existantes, et bien choisir à quel moment lancer l'événement.
- faire en sorte que le jeu se déroule automatiquement, c'est-à-dire en respectant l'ordre suivant :
  - un joueur est renseigné que c'est son tour de jouer,
  - il joue son coup,
  - une pièce est insérée
  - si cette insertion n'est pas bonne (case occupée ou en dehors des limites), le même joueur est renseigné qu'il doit jouer,
  - si l'insertion est bonne, la fin du jeu est testée, puis le joueur suivant est renseigné qu'il doit jouer si le jeu n'est pas terminé.

  Vous pourrez choisir par exemple de réécrire la méthode `InsertPiece`, et dans cette méthode, d'appeler les autres méthodes pour garantir l'ordre de ces appels. Il peut être d'autant plus judicieux de réécrire `InsertPiece` que `Player.Play` appelle cette méthode virtuelle.

Attention, beaucoup de choses sont déjà réalisées dans la bibliothèque `giMorpionCore.dll`. Étudiez la documentation `html` fournie pour en déduire le code à écrire et éviter de réécrire ce qui existe déjà. Apportez également une attention particulière au lancement des événements.

Un rappel utile : si vous réécrivez une méthode virtuelle `MethodA` dans une classe fille, vous pouvez appeler la méthode virtuelle `MethodA` correspondante de la classe mère à l'aide du mot clé `base` : `base.MethodA(...)`.

Conseil : respectez au maximum le pattern standard des événements.

QUESTION 2 : APPLICATION CONSOLE

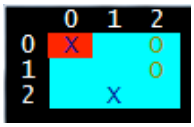
Vous devez réaliser une application Console qui s'abonnera aux 4 événements créés dans la question précédente, et permettra d'afficher le jeu de la manière suivante :

- quand le jeu débute : la grille vide et un message de début

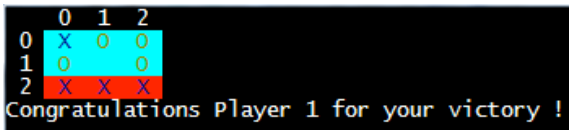
- quand un joueur est renseigné qu'il doit jouer : une phrase lui demandant de jouer

```
Player 0, it's your turn !
```

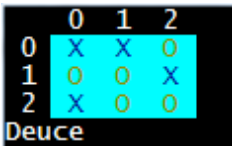
- quand un joueur a inséré une pièce : la grille avec la pièce insérée mise en évidence



- quand la partie est terminée avec un vainqueur : la grille avec la ligne gagnante mise en évidence, et un message au vainqueur,



- quand la partie est terminée sans vainqueur : la grille et un message pour dire qu'il n'y a pas de vainqueur



Conseil important : utilisez la bibliothèque `giMorpionTools.dll` pour l'affichage.

Compléments pour le cours 5

En complément du TP5, voici d'autres exercices sur des thèmes abordés lors du cinquième cours, que je vous conseille de faire afin de vous entrainer davantage. Je vous recommande tout particulièrement de faire les examens des années précédentes (sujets sur l'ENT).

EXERCICE 05_01

Méthodes anonymes

Modifiez le résultat du TP4 en utilisant des méthodes anonymes pour remplacer les méthodes `Linéaire`, `Carré` et `CarréNégatif`.

Recommencez mais en utilisant maintenant des expressions lambda.

EXERCICE 05_02

Méthodes anonymes

Modifiez l'exercice 04_05 en utilisant des méthodes anonymes et supprimer la classe statique `ActionsEtSélections`.

Recommencez mais en utilisant maintenant des expressions lambda.

EXERCICE 05_03

méthodes d'extension

Écrivez les méthodes d'extension suivantes pour des entiers :

- `Abs` : une méthode qui transforme un entier en un sa valeur absolue,
- `Borne` : une méthode qui prend une borne inférieure et une borne supérieure en paramètres, et rend l'entier s'il est compris entre les deux, la borne inférieure si l'entier est inférieur à celle-ci, la borne supérieure si l'entier est supérieur à celle-ci,
- `ToLetter` : une méthode qui rend l'entier en lettres («zéro», «un», «deux»...) si l'entier est compris entre 0 et 9 (inclus) ou «je sais pas» s'il n'est pas compris entre ces bornes.

Testez vos méthodes dans une application Console. Vous pourrez notamment tester l'enchaînement de méthodes d'extension en réalisant par exemple :

```
Console.WriteLine((-13).Abs().Borne(0, 9).ToLetter());
```