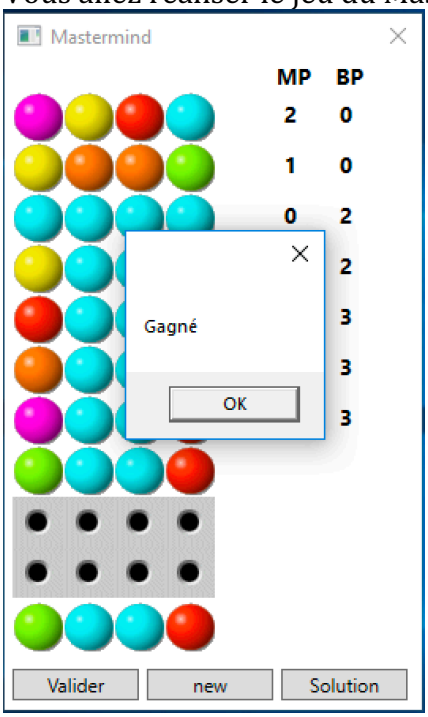
Finir le TD4

Partie 1 : Présentation

Vous allez réaliser le jeu du Mastermind uniquement en code Behind (pas de vue XAML) :



Partie 2 : Préparation du projet

Créez un nouveau projet WPF nommé MasterMind
Créez un répertoire Model et Images.
Importez la classe métier ainsi que les images.

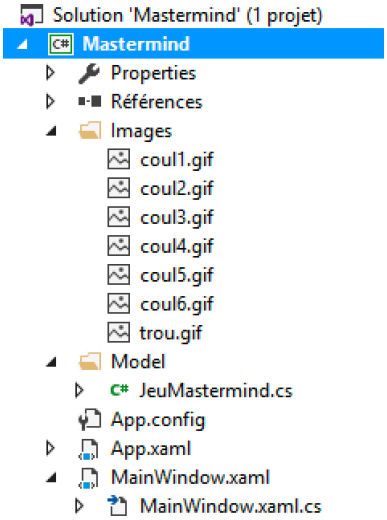
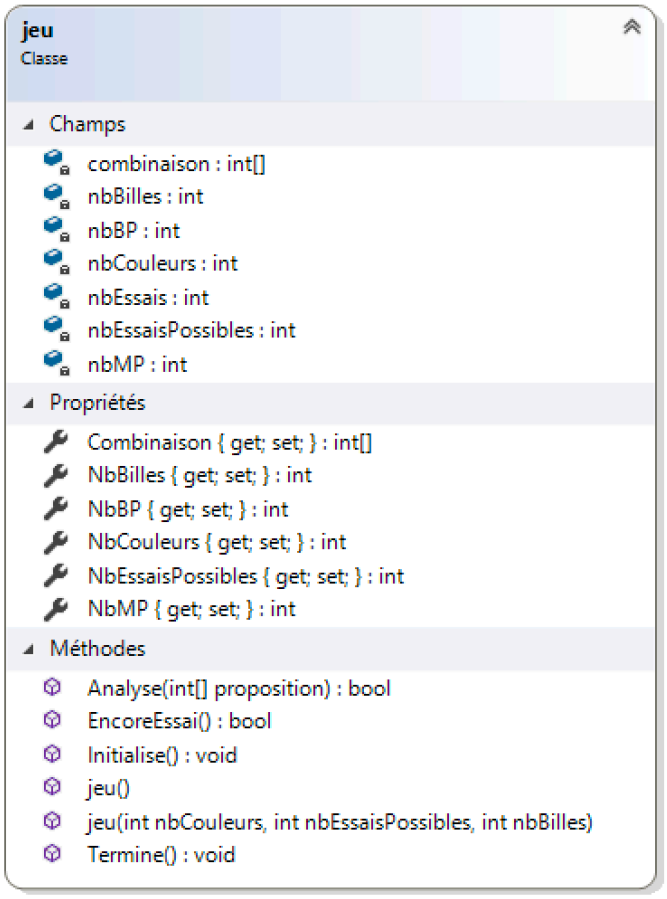


Diagramme de classe de la classe métier :



Analyser la classe métier.

Partie 3 : Réalisation de l'IHM

Vous allez réaliser dynamique la partie IHM en utilisant uniquement du code C# (Behind)
Exception (dans un premier temps) : Modifiez la fenêtre principale en XAML pour avoir une dimension de 280x450px non redimensionnable et centrée.

Créez dynamiquement une grille (champ privé à créer) :
grille = new Grid() ;
Créez deux lignes dans la grille (ligne 1 : boutons et textes, ligne 2 : boutons Valider, New, Solution) :

```
RowDefinition gridRow1 = new RowDefinition();
gridRow1.Height = new GridLength(380);
RowDefinition gridRow2 = new RowDefinition();
gridRow2.Height = new GridLength(50);
grille.RowDefinitions.Add(gridRow1);
grille.RowDefinitions.Add(gridRow2);
this.Content = grille;
```

Créez dynamiquement les 40 boutons à l'aide d'un tableau de boutons (champ privé à créer).
Utilisez la propriété Margin et la classe Thickness pour définir la marge des boutons.
Utilisez la propriété HorizontalAlignment et VerticalAlignment pour ancrer les boutons en haut à gauche (HorizontalAlignment.Left et VerticalAlignment.Top)

Définissez l'image des boutons à l'aide de :

```
ImageBrush imgB = new ImageBrush();
BitmapImage img = new BitmapImage(new Uri("pack://application:,,,/Images/trou.gif"));
 imgB.ImageSource = img;
leBouton.Background = imgB;
```

Placez les boutons dans la ligne du haut de la grille :

```
Grid.SetRow(this.leBouton, 0);
this.grille.Children.Add(this.leBouton);
```

Faites de même pour les boutons de la ligne du bas ainsi que le texte « MP BP » (TextBlock)

Partie 4 : Mise en place des gestionnaires d'évènements

Pour mettre en place un événement il suffit d'utiliser la syntaxe suivante :

```
leBouton.Click += OnLeBoutonClick ;
```

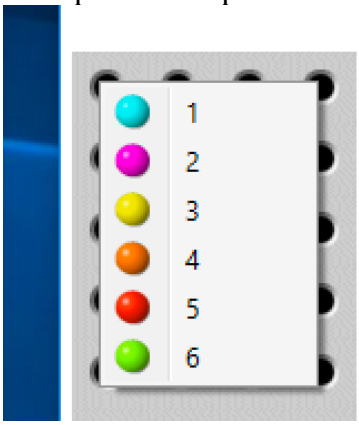
puis définir la méthode OnLeBoutonClick :

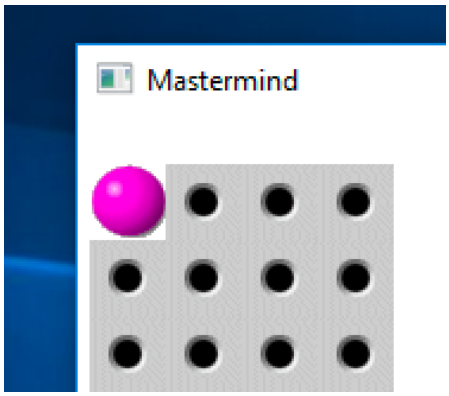
```
private void OnLeBoutonClick(object sender, RoutedEventArgs e)
    {
       // on a appuyé sur le bouton
    }
```

Pour supprimer un gestionnaire d'événement utilisez -=
Mettez en place les gestionnaires d'évènement pour l'ensemble des boutons (attention pour les boutons seule la ligne courante de jeu sera active)

Partie 5 : Mise en place du menu de choix des boutons

Lorsque l'on clique sur un bouton vide un menu contextuel apparait pour choisir le bouton :





Créez dynamique ce menu (listeBilles) au clic sur le bouton (profitez pour mémoriser le bouton cliqué boutonClic = (Button)sender) :
Utilisez la classe ContextMenu et MenuItem.
Pour ajouter un icone utilise la classe Image.
Ajoutez des gestionnaires d'événement sur les items
Pour afficher le menu contextuel utilisez :

```
listeBilles.PlacementTarget = sender as Button;
listeBilles.IsOpen = true;
```

Pour changer l'image du bouton cliqué, inspirez-vous de la partie 3 en utilisant le bouton mémorisé ainsi que le Header du menu contextuel : ((MenuItem)sender).Header

Partie 6 : utilisation de la classe métier

Utilisez la classe métier pour implémenter le cœur du jeu. Vous pourrez définir un tableau pour les réponses.

Remarque :
N'oubliez pas que valider ne doit pas permettre de modifier les essais précédents (supprimez les gestionnaires d'événements précédent).
La ligne active doit changer à chaque fois que l'on valide (une seule ligne active à chaque fois).

Partie 7 : Pour les plus rapides

Mettez en place un menu afin de de choisir la difficulté du jeu (qui varie en fonction du nombre de billes, de couleurs et d'essais).
Modifiez dynamiquement la taille de fenêtre en fonction de la difficulté du jeu.