INF1015 - Programmation orientée objet avancée
Travail dirigé No. 6

Exceptions, espaces de nom, variables de classe

Objectifs :	Permettre à l'étudiant de se familiariser avec les derniers concepts vus dans le cours qui pourraient ne pas avoir été utilisés dans votre conception du projet.
Durée :	Deux semaines de laboratoire.
Remise du travail :	Avant 23h30 le jeudi 21 avril 2022.
Documents à remettre :	sur le site Moodle des travaux pratiques, vous remettrez l'ensemble des fichiers .cpp et .hpp compressés dans un fichier .zip en suivant la procédure de remise des TDs.

Directives particulières

- Ce TD s'intègre dans le projet mais évalue des points précis. Vous remettez simplement une version de votre projet qui inclut les points indiqués ci-dessous. Pour cette remise, vous pouvez commenter/éliminer des parties du projet qui ne compilent pas ou causent des erreurs, de manière à avoir au moins les points indiqués ci-dessous fonctionnels.
- Il est interdit d'utiliser les variables globales; les constantes globales sont permises.
- Vous devez éliminer ou expliquer tout avertissement de « build » donné par le compilateur (avec /W4).
- Respecter le guide de codage, les points pertinents pour ce travail sont donnés en annexe à la fin de l'énoncé du projet.
- N'oubliez pas de mettre les entêtes de fichiers (guide point 33).

Travail à effectuer :

1. Avoir au moins deux « namespace » différents et n'utilisez pas un « using namespace » global sur ces espaces de noms (vous avez le droit de faire des « using » locaux aux méthodes). Nous proposons de mettre votre modèle de données (la gestion du mouvement des pièces du jeu) et l'interface graphique dans deux « namespace » différents.

2. Avoir un compteur d'instances pour la pièce « roi » à l'aide d'une variable de classe, qui s'assure qu'il n'y a jamais plus de 2 rois (normalement un de chaque couleur, mais on ne demande pas de vérifier la couleur). Une exception est lancée s'il y a plus de deux instances de cette classe. Sur réception de cette exception (pas à l'émission de l'exception), un message doit être affiché (le message peut être textuel ou en Qt, à votre choix) et l'exécution continuer à un endroit qui fait du sens (pas avec plus de 2 rois). Si la réception de l'exception se fait dans la même fonction que son émission, vous avez un problème de conception avec séparation du modèle et de la vue.

3. Faire une classe implémentant le RAII pour qu'une fonction utilisant cette classe puisse mettre temporairement une pièce à un endroit sur l'échiquier, et que la pièce s'enlève automatiquement à la destruction de l'instance.

Annexe : Points du guide de codage à respecter

Les points du **guide de codage** à respecter **impérativement** pour ce TD sont :
(voir le guide de codage sur le site Moodle du cours pour la description détaillée de chacun de ces points)

Points ajoutés depuis le TD précédent :

6 : noms des « namespace » minuscules
20 : variables pour composantes de l'interface graphique suffixées par le type d'élément en anglais ou préfixées en français
26 : utiliser des noms complémentaires pour des entités complémentaires
30 : préférer les "enum class" aux anciennes énumérations
31 : classes d'exception avec nom « Exception » comme suffixe en anglais ou préfixe en français
40 : fichiers d'entête avec garde d'inclusion multiple
41 : grouper les `#include`
43 : types locaux devrait être cachés (déclarés dans un seul fichier, ou protégés dans une classe)
64 : utiliser `nullptr` pour un pointeur nul, plutôt que `0` ou `NULL`

Plus les mêmes points que le TD3 :

2 : noms des types en `UpperCamelCase`
3 : noms des variables en `lowerCamelCase`
5 : noms des fonctions/méthodes en `lowerCamelCase`
7 : noms des types génériques, une lettre majuscule ou nom référant à un concept
8 : préférer le mot `typename` dans les template
15 : nom de classe ne devrait pas être dans le nom des méthodes
21 : pluriel pour les tableaux (`int nombres[];`)
22 : préfixe *n* pour désigner un nombre d'objets (`int nElements;`)
24 : variables d'itération `i`, `j`, `k` mais jamais `l`, pour les indexes
27 : éviter les abréviations (les acronymes communs doivent être gardés en acronymes)
29 : éviter la négation dans les noms
33 : entête de fichier
42 : `#include` au début
44,69 : ordonner les parties d'une classe public, protected, private
46 : initialiser à la déclaration
47 : pas plus d'une signification par variable
48 : aucune variable globale (les constantes globales sont tout à fait permises)
50 : mettre le & près du type
51 : test de `0` explicite (`if (nombre != 0)`)
52, 14 : variables vivantes le moins longtemps possible
53-54 : boucles `for` et `while`
58-61 : instructions conditionnelles
62 : pas de nombres magiques dans le code
67-78, 88 : indentation du code et commentaires
83-84 : aligner les variables lors des déclarations ainsi que les énoncés
85 : mieux écrire du code incompréhensible plutôt qu'y ajouter des commentaires