INF1015 - Programmation orientée objet avancée
Travail dirigé No. 6

Bibliothèque de structures de données et algorithmes,
Introduction aux interfaces graphiques
Gestion des exceptions

Objectifs :	Continuer à explorer les différents conteneurs, itérateurs, algorithmes, se familiariser aux interfaces graphiques et à la programmation par évènements, et utiliser la gestion des exceptions.
Durée :	Une séance de laboratoire.
Remise du travail :	Avant 23h59 le dimanche 19 juin 2022.
Travail préparatoire :	Avoir installé Qt, version « open source license », avec Qt Designer et possiblement Qt Creator (community).
Documents à remettre :	Sur le site Github Classroom, vous remettrez l'ensemble des fichiers en suivant la procédure de remise des TDs.

Directives particulières

- Avant même de débuter le TD, vous devez installer Qt. La création d'un compte Qt sera nécessaire bien que la version « open source license » avec Qt Creator community soit gratuite.
- Vous pouvez faire un projet Visual Studio (prend un plugin Qt pour Visual Studio et il y a un exemple de projet sur Moodle) ou Qt Creator. L'éditeur et le débogueur de Visual Studio sont meilleurs mais l'intégration des outils d'édition Qt est plus directe dans Qt Creator.
- Vous pouvez ajouter d'autres fonctions/méthodes et structures/classes, pour améliorer la lisibilité et suivre le principe DRY (Don't Repeat Yourself)
- Il est interdit d'utiliser les variables globales; les constantes globales sont permises
- Respecter le guide de codage, les points pertinents pour ce travail sont donnés en annexe à la fin
- N'oubliez pas de mettre les entêtes de fichiers (guide point 33)

Votre travail consiste à créer une petite application graphique simulant une caisse enregistreuse. Voici les points à respecter :

VUE

1. Deux widgets de type Line Edit permettant de saisir du texte
    a. Un pour la description textuelle d'un article, exemple Sac de chips.
    b. Un pour le prix de l'article saisi en a), exemple 1.99.
2. Trois widgets de type Push Button
    a. Un pour ajouter l'article saisi en 1. dans votre modèle et votre vue.
    b. Un pour retirer de votre modèle et votre vue un/des articles sélectionnés parmi ceux déjà ajoutés.
    c. Un pour tout réinitialiser (on débute une nouvelle commande).
    d. Les 3 boutons émettent un signal lorsque l'on clique dessus. Assurez-vous de lier à ces signaux l'action appropriée. De plus, lorsque votre modèle ajoute ou retire un article, il doit informer la vue pour que celle-ci reflète le changement de la commande d'articles. Pour résumer, le bouton informe le modèle, ensuite le modèle informe la vue. N'oubliez pas que le modèle ne connait pas la vue, par conséquent assurez-vous d'avoir extrait toute l'information nécessaire de votre vue afin de passer celle-ci à votre modèle.

3. Un widget de type Check Box permettant de spécifier si l'article saisi est taxable ou non avant de l'ajouter.
4. Un widget de type List Widget (et non List View) affichant à l'écran tous les articles déjà ajoutés (la commande)
    a. L'affichage d'un article se fait selon ce format : description \t prix \t taxable (si oui, autrement omis).
5. Trois widgets de type Label pour afficher les différents totaux
    a. Un pour le total avant taxes.
    b. Un pour le total des taxes (taux global de 14,975%).
    c. Un pour le total à payer.
6. Tous vos montants doivent être affichés avec seulement deux chiffres après la virgule, exemple 14.975 = 14.98.
7. Des étiquettes pour bien identifier vos différents champs (Description article, Prix article, Sous-total, etc.)
8. Le bouton pour retirer un/des articles sélectionnés doit être désactivé si la commande ne contient pas d'articles

MODÈLE

1. Une struct Article ayant les champs suivants (pas besoin d'encapsulation, getters/setters)
    a. string description
    b. double prix
    c. bool taxable
2. Un conteneur STL approprié pour y contenir tous les articles ajoutés.
3. Un attribut pour le total avant taxes.
4. Pour le montant total des taxes, vous devez utiliser une fonction lambda pour le calcul et un algorithme STL pour parcourir tous vos articles ajoutés dans votre vecteur. Il est interdit d'utiliser un for !
5. Pour retirer un article, vous devez utiliser un itérateur afin de le trouver dans votre vecteur, il est interdit d'utiliser un index !
6. Votre modèle doit lancer une exception, capturée par la vue pour afficher un message, lorsque la description est vide ou lorsque le prix est zéro.

IMPORTANT

Si vous utilisez « move », attention qu'il y a aussi un « move » pour déplacer un widget Qt, vous pouvez avoir à écrire « std::move »

Si vous utilisez Visual Studio, il y a un document sur Moodle pour l'installation de Qt, l'extension Qt pour Visual Studio et comment faire la configuration pour s'assurer que ça fonctionne.

Si vous utilisez Qt Creator, veuillez vérifier que votre projet utilise la version 2020 de C++. La ligne 5 de votre fichier .pro devrait être la suivante : `CONFIG += c++latest`

Annexe 1 : 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)

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