INF1010

Programmation Orientée-Objet

Travail pratique #6

Interfaces Utilisateurs et exceptions

Objectifs :	Permettre à l'étudiant de se familiariser avec les concepts d'interfaces utilisateurs, d'exceptions et de variables de classe en C++.
Remise du travail :	Jeudi 14 avril avant 8h
Références :	Notes de cours sur Moodle + http://doc.qt.io/
Documents à remettre :	Fichiers .cpp et .h sous la forme d'une archive .zip.
Directives :	Directives de remise des Travaux pratiques sur Moodle Les en-têtes et les commentaires sont obligatoires. Veuillez suivre le guide de codage

Travail à réaliser

Au cours des laboratoires précédents, vous avez développé un programme de gestion de bibliothèque assez élaboré. Il est maintenant temps d'y intégrer une interface graphique.

Une interface simple permettant de visualiser et d'ajouter des emprunts aux abonnés est presque terminée. Votre travail consistera à la gestion de **quelques signaux d'événements et d'exceptions**. L'interface fonctionnelle doit ressembler aux images ci-dessous. Sauf avis contraire, toutes les classes et fonctions sont fournies et terminées. Les sections à faire ou à modifier sont clairement indiquées dans le fichier fournis par le commentaire **« !!!!!! A FAIRE !!!!!! »**.

Pour l'affichage des différents éléments, vous pouvez vous référer aux différentes images qui vous sont fournies dans la partie « Exemples d'exécution ».

Dans la liste des fichiers fournis, le fichier .pro est le projet Qt, il suffit de l'ouvrir avec Qt Creator.

N'oubliez pas d'exécuter « qmake » après l'ouverture du projet Qt et après l'ajout de nouveaux fichiers.

Classe *Bibliotheque*

Cette classe reste inchangée, sauf les méthodes *emprunter()* et *retourner()* qui vont désormais lancer des exceptions lors d'un échec d'emprunt ou de retour (plutôt que de renvoyer une valeur booléenne).

Le code de ces méthodes vous est fourni. Il ne reste qu'à ajouter les tests des conditions d'emprunt/retour et à lancer les exceptions avec les messages appropriés.

- La méthode *emprunter()* doit lancer une exception de type *ExceptionEchecEmprunt* dans les cas suivants :

Condition	**Message**
ab == nullptr	Abonné non trouvé
obj == nullptr	Objet empruntable non trouvé
obj->obtenirNbDisponibles() == 0	Aucun exemplaire disponible
ab->obtenirAge() < obj->obtenirAgeMinimal()	L'abonné n'a pas l'âge minimal requis
estTrouve == true	L'abonné a déjà emprunté cet objet
count >= ab->obtenirLimiteEmprunt()	L'abonné a atteint sa limite d'emprunts

- La méthode *retourner()* doit lancer une exception de type *ExceptionEchecRetour* lorsque l'emprunt correspondant au matricule et à la cote passés en paramètres n'est pas trouvé.

Classe *ExceptionEchecRetour*

Écrire une classe d'exception simple *ExceptionEchecRetour* qui hérite de l'exception standard *std::runtime_error*. Cette exception personnalisée sera lancée lors d'un échec de retour dans la méthode *retourner()* de la classe *Bibliotheque*.

Classe *ExceptionEchecEmprunt*

Écrire une classe d'exception *ExceptionEchecEmprunt* qui hérite de l'exception standard *std::runtime_error*. Cette exception personnalisée sera lancée lors d'un échec d'emprunt dans la méthode *emprunter()* de la classe *Bibliotheque*.

Cette classe doit contenir l'attribut suivant :

- *compteur_* : un entier (int) statique privé permettant de compter le nombre de fois que l'exception *ExceptionEchecEmprunt* est lancée.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par paramètre recevant une chaîne de caractères (string) contenant le message de l'erreur. Le constructeur doit incrémenter la valeur du compteur.
- La méthode statique public *obtenirValeurCompteur()* qui retourne la valeur de *compteur_*.

Classe *DialogEmprunter*

Cette classe hérite de *QDialog* et permet d'afficher une nouvelle fenêtre permettant de sélectionner un objet à emprunter lorsque le bouton Ajouter emprunt est cliqué. **Cette classe vous est fournie et n'a pas à être modifiée.**

Classe *MainWindow*

La classe *MainWindow* est la fenêtre principale de l'application. Elle hérite de la classe *QMainWindow* et comprend les attributs suivants :

- Un pointeur vers une bibliothèque
- Un *QListWidget** : pour afficher la liste des abonnés
- Un *QTableView** : pour afficher les emprunts d'un abonné
- Un *QStandardItemModel** : modèle de données contenant les emprunts d'un abonné
- Deux *QPushButton** : pour les boutons Retourner et Ajouter emprunt

Seules les méthodes avec un crochet √ doivent être modifiées :

- Un constructeur par paramètres, recevant entre autres un pointeur de bibliothèque, qui initialise l'interface.
- √ Une méthode privée *setUI()* qui crée les différents éléments de l'interface. Vous devez modifier cette fonction pour ajouter les boutons Retourner et Ajouter emprunt à l'interface, tel que montré en exemple plus bas. *(voir Figure* 1)
- √ Une méthode privée *setConnections()* qui connecte les évènements des éléments de l'interface aux méthodes *private slots*. Trois connexions doivent être ajoutées :
    - Un clic du bouton Ajouter emprunt doit engendrer un appel à la méthode *ajouterEmprunt()* de la classe *MainWindow*;

    - Un clic du bouton Retourner doit engendrer un appel à la méthode *retirerEmprunt()* de la classe *MainWindow*;
    - Une connexion entre le signal *empruntAjoute()* et la méthode *afficherEmpruntsAbonne()* de la classe *MainWindow*;
- Une méthode privée *chargerAbonnes()* qui remplit l'objet *QListWidget* avec la liste de tous les abonnés de la bibliothèque.
- Un signal *empruntAjoute(),* qui reçoit en paramètre un pointeur sur un *QListWidgetItem* (un élément sélectionné dans la liste d'abonnés).
- Une méthode (slot) privée *afficherEmpruntsAbonne()* qui récupère le pointeur de l'abonné sélectionné dans la liste et les affiche dans la *QTableView*.
- √ Une méthode (slot) privée *ajouterEmprunt()* qui affiche dans une fenêtre dialogue la liste des objets empruntables et permet d'en sélectionner un à emprunter. *(voir Figure 2)*
    - Cette méthode est sujette à deux types d'exceptions ! **Voir les indications dans les commentaires de la méthode**. Vous devez donc ajouter le code nécessaire pour intercepter les exceptions le plus spécifiquement possible et afficher un *QMessageBox* contenant le message d'erreur tel que montré en exemple à la fin de l'énoncé. Dans le cas d'un échec d'emprunt, le titre du *QMessageBox* doit également contenir le nombre de fois que ce type d'erreur survient (ex : « Echec d'emprunt (4e erreur) »). *Indice : utiliser le compteur de la classe d'exception. (voir Figure 4)*
    - Lorsqu'un emprunt est réussi, le signal *empruntAjoute()* doit être émis afin de permettre la mise à jour automatique du tableau d'emprunts de l'abonné. Aucun signal ne doit être émis en cas d'échec d'emprunt. *(voir Figure 3)*
- √ Une méthode (slot) privée *retirerEmprunt()* dont l'implémentation restera inachevée dans ce TP. Pour les fins de l'exercice, la fonction tente de retourner un objet empruntable inexistant pour un abonné inexistant. Une exception sera donc inévitablement lancée par la fonction *retourner()* de la classe *Bibliotheque*. Vous devez intercepter cette exception le plus spécifiquement possible et afficher un *QMessageBox* avec un message approprié, tel que montré en exemple. *(voir Figure 6)*
- Une méthode *obtenirAbonneSelectionne()* qui retourne l'élément sélectionné dans la liste d'abonnés.

Exemples d'exécution

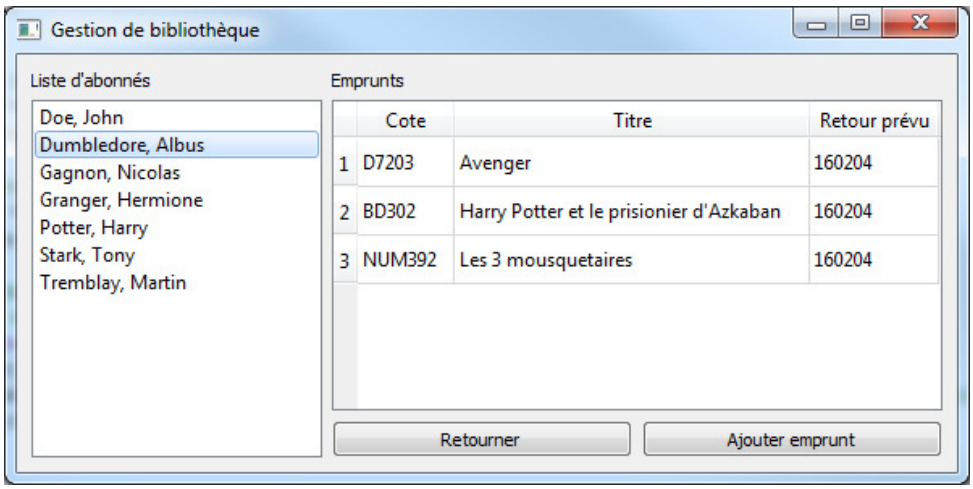

Figure 1. Interface incluant les boutons Retourner et Ajouter emprunt.

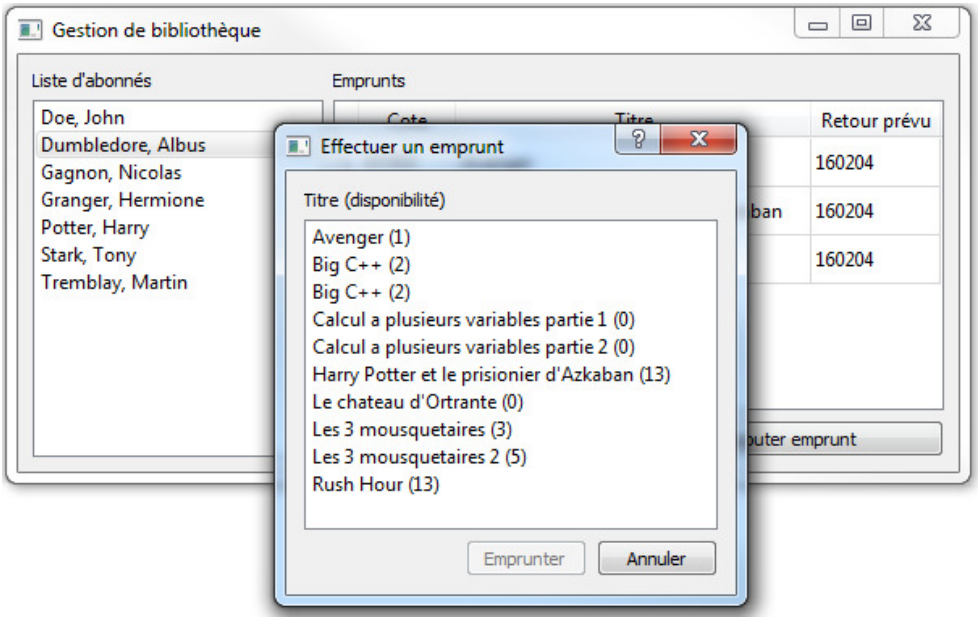

Figure 2. Nouvelle fenêtre dialogue s'affichant lorsqu'on clique sur Ajouter emprunt.

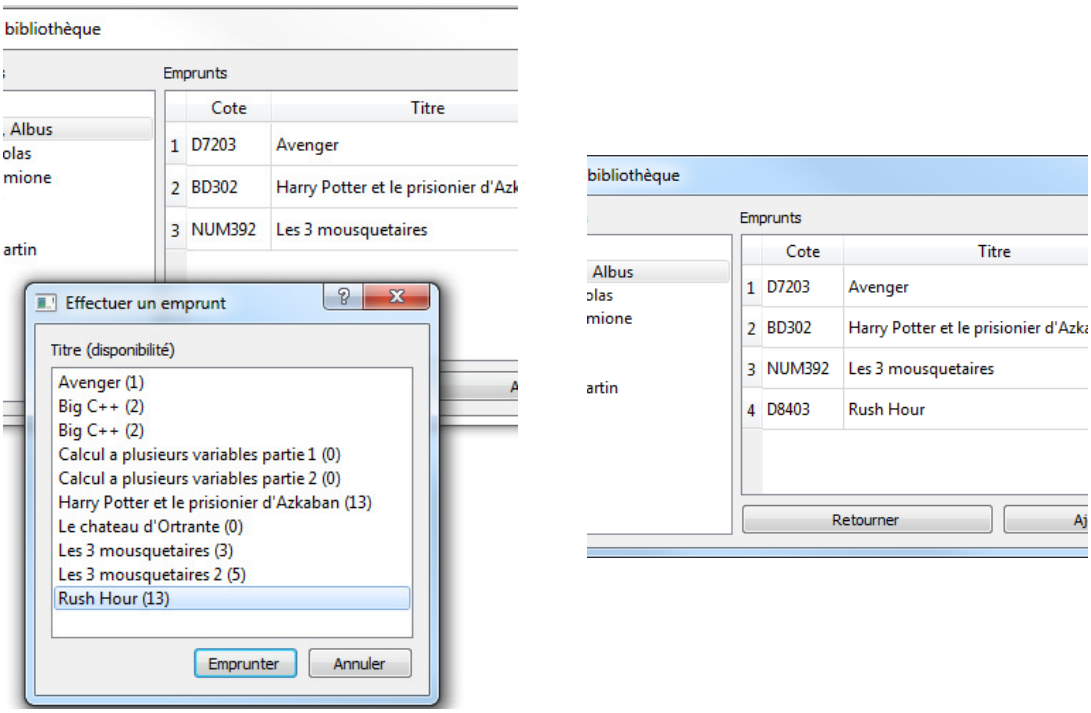

Figure 3. Exemple d'ajout d'un emprunt de Rush Hour pour l'abonné Albus. À droite, la liste d'emprunts a été mise à jour automatiquement après avoir cliqué sur le bouton Emprunter de la fenêtre dialogue.

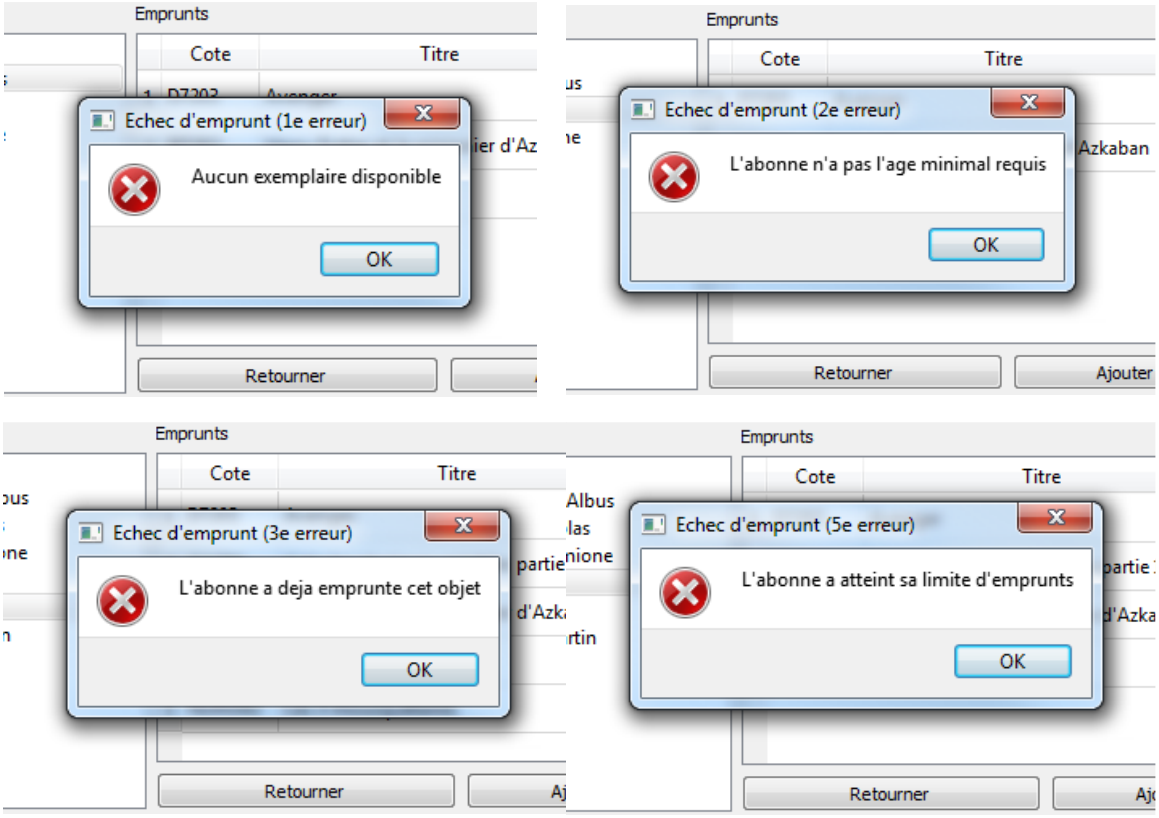

Figure 4. Exemples de messages d'erreurs affichés lors d'un échec d'emprunt.

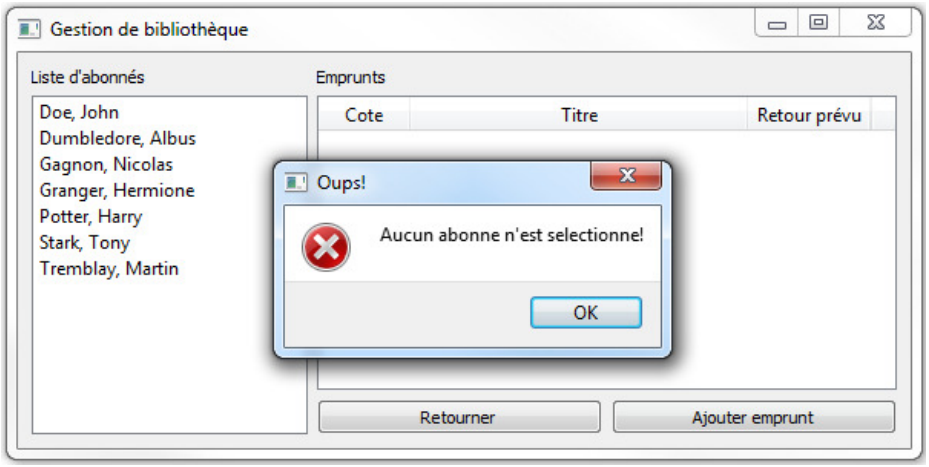

Figure 5. Message d'erreur affiché lorsqu'on clique sur le bouton Ajouter emprunt, mais qu'aucun abonné n'a été sélectionné.

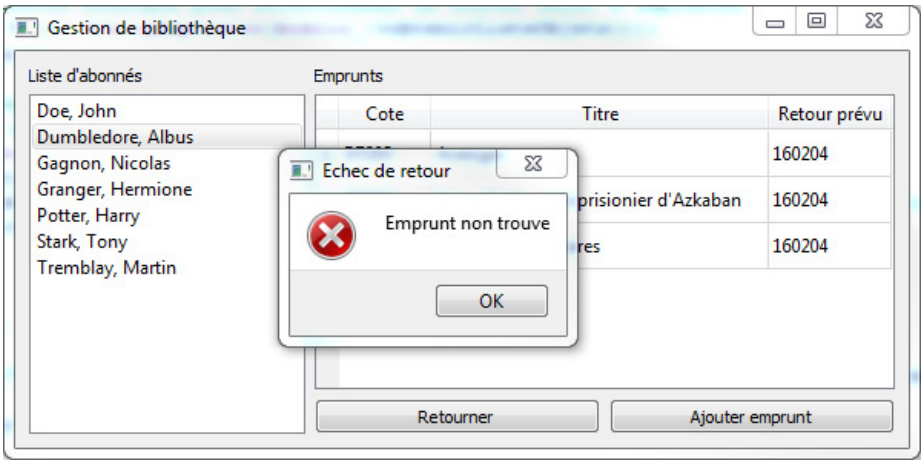

Figure 6. Message d'erreur affiché lorsqu'on clique sur le bouton Retourner.

Correction

La correction du TP6 se fera sur 20 points. Voici les détails de la correction :

- (3 points) Compilation du programme;
- (3 points) Exécution du programme;
- (4 points) Comportement exacte des méthodes et de l'interface;
- (1 point) Utilisation adéquate des Widgets Qt (boutons, boîtes de message)
- (3 points) Utilisation adéquate des signaux et des connexions en Qt;
- (3 points) Gestion adéquate des exceptions;
- (2 point) Utilisation adéquate des éléments statiques de classe;
- (1 point) Documentation du code;

Pour partir du bon pied

La documentation officielle de Qt est votre meilleur ami, vous pouvez la retrouvez à l'adresse :

http://qt-project.org/doc/

Tapez simplement le nom de votre Widget, slot ou signal… dans la zone de recherche « Search doc » et le site vous renvoie directement vers le lien menant à la description de votre recherche.

Environnement de programmation

L'environnement de programmation pour ce TP se fait sur Qt Creator et non sur MSVC.

Pour lancer votre environnement de programmation Qt Creator, s'il n'apparait pas dans le menu Démarrer->tous les programmes.

Allez sur le bureau, vous trouverez le fichier nommé Logiciels, dans ce fichier chercher Qt ou QtSDK, entrez-y et cliquez sur le raccourcis de Qt qui s'y trouve.

Création d'un nouveau projet Qt avec QtCreator

Suivez les étapes suivantes :

1) Cliquez sur l'onglet « File », ensuite « New File or Project »,
2) Choisissez l'option « Other Project » dans l'onglet « Project ».
3) Choisissez dans la fenêtre de droite « Empty Qt Project »
4) Cliquez sur « choose ».
5) Entrez le nom de votre projet en faisant « next » et en laissant les paramètres par défaut fournis.
6) Une fois que votre projet est créé, cliquez une seconde fois sur « File », « New File or Project », cette fois ci choisissez « C++ » dans l'onglet « File and Classe » et créer le type de fichier que vous voulez : entête (.h) ou source (.cpp) ou qt.

À partir de ce moment, tout se passe comme sous Visual Studio.

Ouverture d'un projet existant

1) Cliquez sur l'onglet « File », ensuite « Opne File or Project »,
2) Ouvrir le projet ayant l'extension .pro

Ajout des fichiers déjà existants :

Vous pouvez directement importer les fichiers avec *Add Existing Files*.

3) Une fois votre projet crée avec la procédure énoncée ci-haut, faites un clic droit sur le nom de votre projet.
4) Choisissez l'option *Add existing file* et vous pouvez importer directement vos fichiers à partir de l'emplacement où vous les avez sauvegardés. Là aussi les choses se passent comme avec Visual Studio

Pour compiler votre code :

5) Cliquez sur *Debug* ensuite sur *Run* ou encore utilisez simplement le raccourci clavier Ctrl + R.