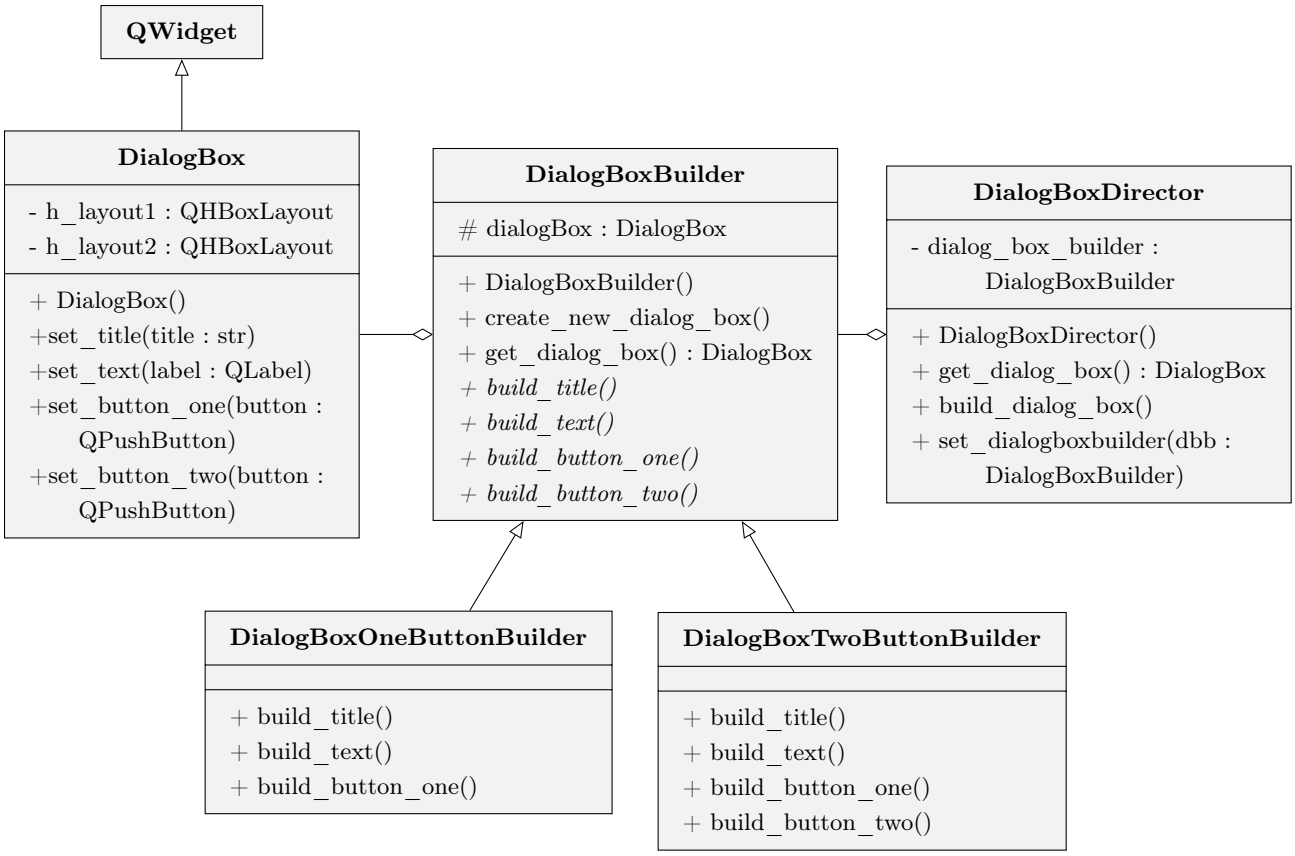
1. But du TP

Implémenter le *design pattern Builder* pour monter une boite de dialogue ayant un ou deux boutons. Ici chaque *Builder* définira cinq méthodes qui permettront au *Director* de construire la boite de dialogue. Ainsi, celui-ci pourra construire une boite dialogue en s'appuyant sur un *Builder* particulier.

2. Diagramme UML



3. Classe `DialogBox`

Cette classe doit permettre de créer une boite de dialogue (qui hérite de `QWidget`) et de lui ajouter, éventuellement, un titre, un texte et deux boutons. Le constructeur devra initialiser les deux attributs stockant des layout horizontaux et créer un layout vertical qui les englobera. Les

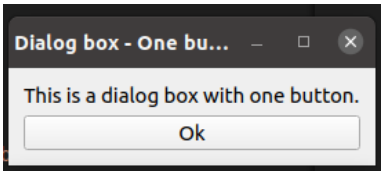
quatre méthodes `set...()` aurront respectivement pour but de modifier le titre de la fenêtre, d'ajouter le texte ou les boutons passés en paramètre à la fenêtre.

4. Classe `DialogBoxBuilder`

Cette classe permet de définir les méthodes `create_new_dialog_box()` et `get_dialog_box()`. `create_new_dialog_box()` doit initialiser l'attribut `dialog_box` tandis que la méthode `get_dialog_box()` doit le renvoyer. Les autres méthodes contiennent un code vide car leur implémentation concrète se fera à travers les classes `DialogBoxOneButtonBuilder` et `DialogBoxTwoButtonsBuilder`.

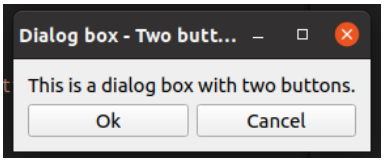
5. Classe `DialogBoxOneButtonBuilder`

Cette classe doit permettre de créer une boite de dialogue contenant un titre, un texte et un bouton. Pour cela elle définit les méthodes `build...()` qui appellent les méthodes `set...()` de `DialogBox` avec des éléments graphique prédéfinis pour paramétrer une nouvelle boite de dialogue. Votre interface pourra ressembler à la figure ci dessous.



6. Classe `DialogBoxTwoButtonsBuilder`

Cette classe doit permettre de créer une boite de dialogue contenant un titre, un texte et deux boutons. Comme précédemment, elle définit les méthodes `build...()` qui appellent les méthodes `set...()` de `DialogB1ox` avec des éléments graphique prédéfinis pour paramétrer une nouvelle boite de dialogue. Votre interface pourra ressembler à la figure ci dessous.



7. Classe `DialogBoxDirector`

Cette classe permet de construire une nouvelle `DialogBox` en s'appuyant sur un `Builder` particulier. En effet, la méthode `set_dialogboxbuilder()` doit permettre de définir le `Builder` à utiliser par le `Director`. Ainsi la méthode `build_dialog_box()`pourra appeler successivement les méthodes du builder, à savoir :

— `create_dialogbox()`

— `build_title()`
— `build_text()`
— `build_button_one()`
— `build_button_two()`

8. Classe `Main`

Écrivez un *main* du programme. Celui-ci devra instancier une `DialogBoxDirector` permettant de créer deux boites de dialogues correspondant au deux `Builder` écrit. Par exemple pour la boite de dialogue à un bouton, on aura :

Programme principal

```
# We create the director.
dialog_box_director: DialogBoxDirector = DialogBoxDirector()

# We create a one button dialog box.
one_button_builder: DialogBoxOneButtonBuilder = DialogBoxOneButtonBuilder()
dialog_box_director.set_dialog_box_builder(one_button_builder)
dialog_box_director.build_dialog_box()
dialog_box: DialogBox = dialog_box_director.get_dialog_box()
dialog_box.show()
```

9. Pour aller plus loin

Posez vous la question suivante : Que faudrait-il faire pour généraliser le code pour créer une boite de dialogue avec nboutons ?

Toutes les méthodes `build...()` ne revoient rien. Une technique intéressante et de faire en sorte que ces méthodes retourne les builders eux-mêmes, permettant au lieu d'utiliser :

```
self.dialog_box_builder.build_title()
self.dialog_box_builder.build_text()
self.dialog_box_builder.build_button_one()
self.dialog_box_builder.build_button_two()
```

d'écrire :

```
self.dialog_box_builder\
    .build_title()\
    .build_text()\
    .build_button_one()\
    .build_button_two()
# ou
self.dialog_box_builder.build_title().build_text().build_button_one().build_button_two()
```

Modifiez votre code en conséquence.