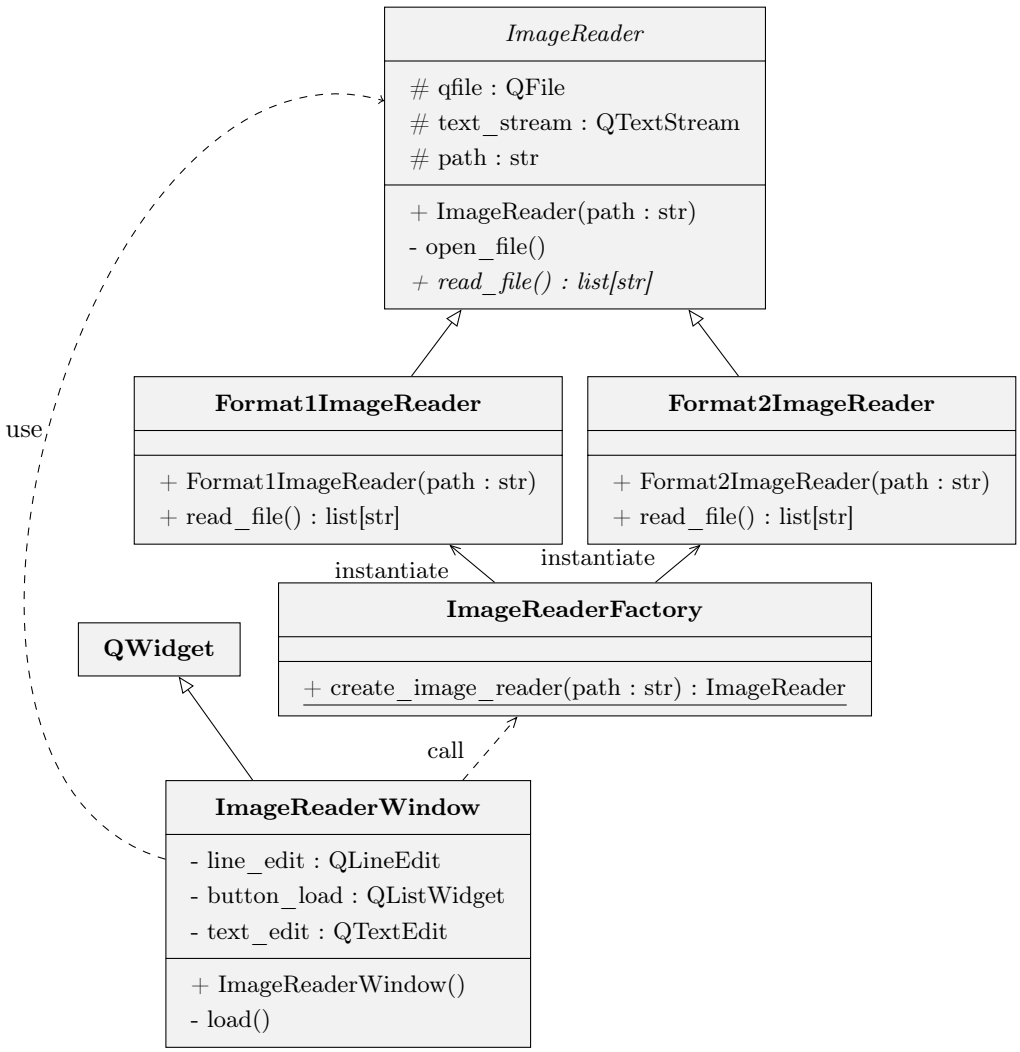
1. But du TP

Implémenter le *design pattern* **Factory** pour créer un lecteur d'image (en mode texte/ascii art) permettant de lire plusieurs formats (*.fm1* et *.fm2*). La Factory permettra de renvoyer automatiquement, et de manière transparente, le lecteur correspondant à l'extension d'image spécifié par l'utilisateur.

2. Diagramme UML



3. Classe ImageReader

C'est une classe abstraite qui doit permettre d'ouvrir un fichier texte (dont le chemin est path) à partir d'un `QFile` et d'un `QTextStream` (tous deux attributs de la classe). Le constructeur doit permettre de mettre à `None` les attributs `qfile` et `textStream`, d'initialiser le `path` et d'ouvrir le fichier grâce à la méthode `open_file()`. Cette méthode doit, quant à elle, ouvrir le fichier texte en instanciant les attributs `qfile` et `text_stream` (voir la doc de QTextStream : `https://doc.qt.io/qtforpython-5/PySide2/QtCore/QTextStream.html#detailed-description`).

4. Classe Format1ImageReader

Cette classe doit permettre de décoder le fichier d'extension *.fm1*. Pour cela, le constructeur de la classe pourra appeler le constructeur de la classe mère en lui passant la variable `path` en argument.

```
data: list[str] = []
# Read the file.
width: int = int(self.text_stream.readLine())
height: int = int(self.text_stream.readLine())
for i in range(height):
    line: str = ""
    for j in range(width):
        line += self.text_stream.readLine()
    data.append(line)
return data
```

5. Classe Format2ImageReader

Cette classe doit permettre de décoder le fichier d'extension *.fm2*. Pour cela, le constructeur de la classe pourra appeler le constructeur de la classe mère en lui passant la variable `path` en argument.

```
# Read the file.
line: str = ""
width: int = int(self.text_stream.readLine())
temp: list[str] = self.text_stream.readLine()
for i in range(len(temp)):
    if i % width == 0:
        data.append(line)
        line = ""
    line += temp[i]
data.append(line)
return data
```

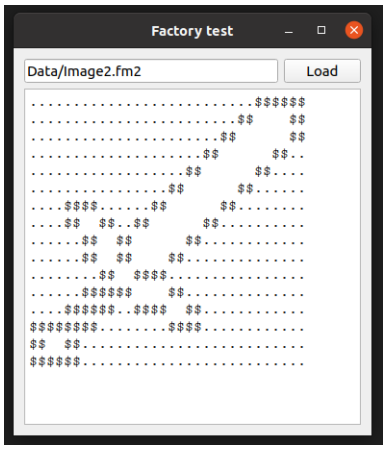
6. Classe `ImageReaderFactory`

Cette classe doit permettre de créer et de renvoyer un `ImageReader` correspondant au format d'image fourni. En effet, si l'extension est *.fm1* alors il doit renvoyer un `Format1ImageReader`, en revanche si l'extension est *.fm2* il doit renvoyer un `Format2ImageReader`. C'est le rôle de la fonction `create_image_reader()`.

7. Classe `ImageReaderWindow`

En vous inspirant des captures d'écran suivantes, créez une classe `ImageReaderWindow` permettant de rentrer le chemin d'une image (voir les ressources fournies avec le TP) et d'afficher son contenu décodé en fonction de son format. Cette interface devra utiliser la méthode statique de `ImageReaderFactory` pour créer automatiquement le lecteur correspondant au chemin entré. Pour un visuel correct, utilisez une police à chasse fixe (chaque caractère a la même taille) avec :

```
self.textEdit.setFontFamily("Courier")
```



8. le `Main`

Écrivez le programme principal qui instanciera une `ImageReaderWindow` permettant de créer et de lancer l'interface graphique.

9. Pour aller plus loin

Que faudrait-il changer dans le code pour prendre en compte un nouveau format de fichier ? Faites en sorte de pouvoir afficher les fichiers au format *.fm3*.