SIM-SysIn : Test Final, partie écrite

(Durée 20 minutes, sans document)

Nom	*Prénom*	*Groupe TD*	*Date*
			15 décembre 2011

1 Soit le fragment de code Python :

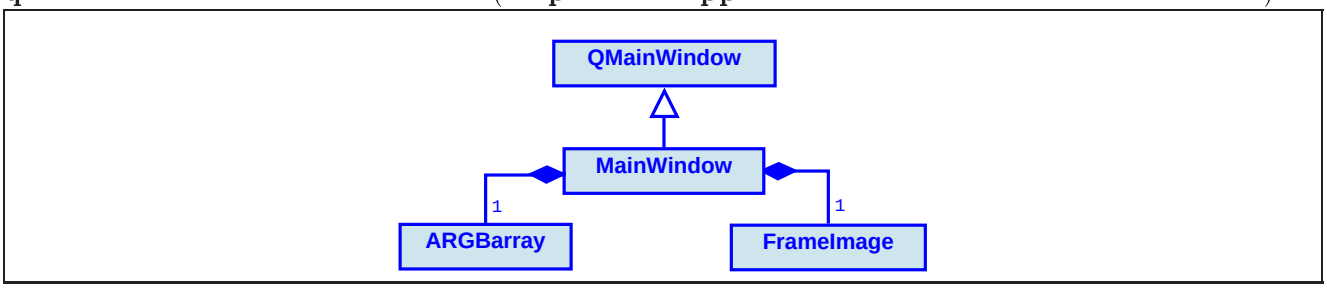
*ModMainWindow.py

```
1  # -*- coding: latin-1 -*-
2  from PyQt4.QtGui import QMainWindow, QPushButton, QImage, QLabel
3
4  from ModFrameImage import FrameImage
5  from ModImageArray import ImageArray
6
7  class MainWindow(QMainWindow):
8      "This is the main window that implements the graphical user interface"
9
10     def __init__(self):
11     # constructor
12         QMainWindow.__init__(self)
13         frame1W, frame1H = 220,220
14         ...
15         spaceH, spaceV = 20,40
16         ...
17         self.__cadre1 = FrameImage(self, spaceH, spaceV, frame1W, frame1H)
18         ...
19         self.__ARGBarray = ImageArray()
20         ...
21         # Menu bar, with sub-menus
22         menubar = self.menuBar()
23         ...
24         menuF = menubar.addMenu('Fichier')
25         ...
26         menuT = menubar.addMenu('Transformer')
27         menuT.addAction('Reinit',      self.__ReinitImage2)
28         menuT.addAction('Negatif',     self.__Negative)
29         ...
```

(les points de suspension remplacent des lignes inutiles vis à vis des questions ci-dessous)

1. Quelle est la ligne de code qui permet de récupérer la définition la classe `QMainWindow` :
   c'est la ligne 2.
2. À quoi sert la ligne 1 :
   à spécifier à l'éditeur IDLE quel est l'encodage des caractères accentués du programme.
3. Où se trouvent les définitions des classes `FrameImage` et `ImageArray` ?
   Respectivement dans les modules `ModFrameImage` et `ModImageArray,` importés ligne 4 et 5.
4. À quoi sert la ligne 7 :
   Elle introduit la définition de la classe `MainWindow`.
5. À quoi sert la ligne 10 :
   Définition du constructeur de la classe.
6. À quoi sert la ligne 12 :
   Elle permet d'exécuter le constructeur de la classe de base `QMainWindow`.
7. À quoi sert la ligne 19 :
   Elle permet de créer un objet de type `ImageArray` en apellant le constructeur de la classe `ImageArray`.

1. Tracer le diagramme UML correspondant strictement au code Python (lignes 7 à 19) en ne montrant **que les relations entre les classes (ne pas faire apparaître les attributs ni les méthodes)** :



2. Sachant que :
   - la ligne 22 crée une barre de menu en haut de la fenêtre de l'application,
   - les lignes 24 et 26 créent dans cette barre les menus `'Fichier'` et `'Transformer'`
   - la ligne 28 ajoute le sous-menu `'Negatif'` au menu `'Transformer'` et l'associe à l'action d'exécuter la méthode privée `Negative` ,

   écrire les lignes de code qui créent le sous-menu `'GradientH'` dans le menu `'Transformer'` et l'associent à la méthode privée `HGradient`

```
menuT.addAction('GradientH',self.__HGradient)
```

2 Soit le fragment de code Python :

```
1    def __InitArrays(self,raw,col):
2        self.__nbR = raw
3        self.__nbC = col
4        self.__R = ndarray((self.__nbR, self.__nbC), dtype=float)
5        self.__G = ndarray((self.__nbR, self.__nbC), dtype=float)
6        self.__B = ndarray((self.__nbR, self.__nbC), dtype=float)
7        self.__A = ndarray((self.__nbR, self.__nbC), dtype=float)
```

1. Décrire et qualifier toutes les variables intervenant dans la méthode privée **`InitArrays`** :
   raw : argument, entier donnant le nombre de lignes
   col : argument, entier donnant le nombre de colonnes
   self.__nbR : attribut privé de type entier, donne le nombre de lignes des tableaux
   self.__nbC : attribut privé de type entier, donne le nombre de colonnes des tableaux
   self.__R : attribut privé de type tableau ndarray, contient l'intensité rouge des pixels
   self.__G : attribut privé de type tableau ndarray, contient l'intensité verte des pixels
   self.__B : attribut privé de type tableau ndarray, contient l'intensité bleue des pixels
   self.__A : attribut privé de type tableau ndarray, contient l'information de transparence des pixels