Université de Pau et des Pays de l'Adour - IUT DE BAYONNE - PAYS-BASQUE année 2022-2023

Département Informatique SAÉ 20.1

Saé 2.01 – Développement d’une application

Lecteur de diaporamas – Dossier d’Analyse et conception

## Compléments de spécifications externes.

*On précise uniquement les points qui vous ont semblé flous ou bien incomplets. Rien de plus à signaler dans cette étude.*

## Scénarios

*Description du scénario nominal et de un / deux scénarios alternatifs afin de mettre en évidence les interactions entre le système et l’utilisateur*

## Diagramme de classe (UML)

1. Le diagramme de classes UML se focalise sur les classes **métier**, cad celles décrivant les éléments structurants de l’application, indépendamment des éléments d’interface.
2. Dictionnaire des éléments pour chaque classe

|  |  |  |  |
| --- | --- | --- | --- |
| **Classe XXXXX** | | | |
| **Nom attribut** | **Signification** | **Type** | **Exemple** |
| xxx | xxx | unsigned int | 1 |
| xxx | xxx | unsigned int | 1 |

Tableau 2 : Dictionnaire des éléments - Classe xxx

1. Dictionnaire des méthodes : vous pouvez fournir directement le fichier entête de chaque classe.

Exemple (classe lecteur de la version Console) :

#ifndef LECTEUR\_H

#define LECTEUR\_H

#include "image.h"

#include <vector>

typedef vector<Image\*> Diaporama; // Structure de données contenant les infos sur les images

class Lecteur

{

public:

Lecteur();

void avancer(); // incrémente \_posImageCourante, modulo nbImages()

void reculer(); // décrémente \_posImageCourante, modulo nbImages()

void changerDiaporama(unsigned int pNumDiaporama); // permet de choisir un diaporama, 0 si aucun diaporama souhaité

void afficher(); // affiche les informations sur lecteur-diaporama et image courante

unsigned int nbImages(); // affiche la taille de \_diaporama

Image\* imageCourante(); // retourne le pointeur vers l'image courante

unsigned int numDiaporamaCourant();

private:

unsigned \_numDiaporamaCourant; // numéro du diaporama courant, par défaut 0

Diaporama \_diaporama; // pointeurs vers les images du diaporama

unsigned int \_posImageCourante; /\* position, dans le diaporama,

de l'image courante.

Indéfini quand diaporama vide.

Démarre à 0 quand diaporama non vide \*/

private:

void chargerDiaporama(); // charge dans \_diaporama les images du \_numDiaporamaCourant

void viderDiaporama(); // vide \_diaporama de tous ses objets image et les delete

};

#endif // LECTEUR\_H

Figure 4 : Schéma de classes = Classe XXX

1. Remarques concernant le schéma de classes
   1. On ne s’intéresse qu’aux attributs et méthodes métier. Notamment, on ne met pas, pour l’instant, ce qui relève de l’affichage car ce sont d’autres objets du programme (widgets) qui se chargeront de l’affichage. Par contre, on n’oublie pas les méthodes getXXX(), qui permettront aux objets métier de communiquer leur valeur aux objets graphiques pour que ceux-ci s’affichent.
   2. On n’a mis ni le constructeur ni le destructeur, pour alléger le schéma.
   3. D’autres attributs et méthodes pourront venir ultérieurement compléter cette première vision ANALYTIQUE de l’application. Il s’agira des attributs et méthodes dits DE CONCEPTION nécessaires au développement de l’application.

# Version v0 – Version console seule

## Implémentation et tests

## 4.1 Implémentation

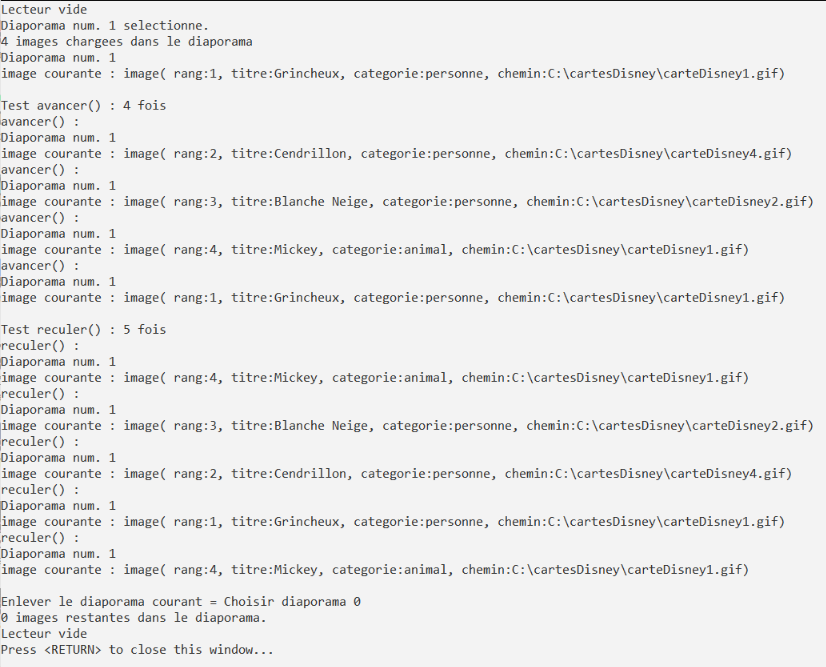
### Liste et rôle des fichiers de cette version :

|  |  |
| --- | --- |
| lecteur.h | Spécification de la classe Lecteur |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Spécification de la classe Image |
| image.cpp | Corps de la classe Image |
| main.cpp | Teste les méthodes de la classe Lecteur |

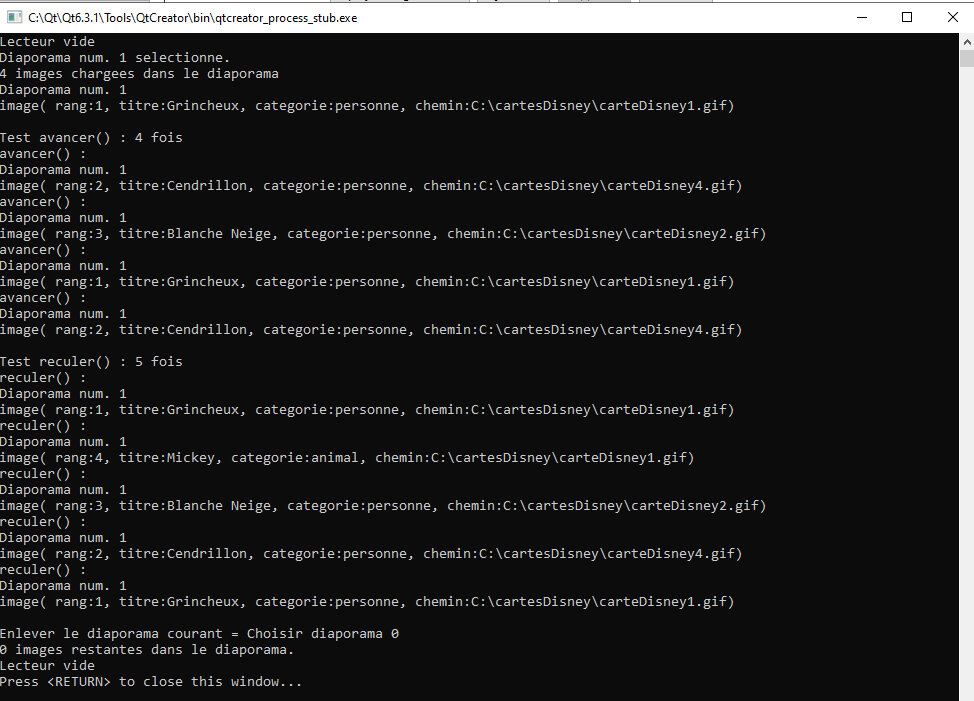
## 4.2 Test

Test avec le programme fourni main.cpp

*Résultat que nous devons obtenir*



*Résultat obtenu après compilation :*



## 4.3 Temps Passé

## Pour la v0 de l’application, nous avons passé aux alentours d’une semaine avec les modifications apportées à la fin. En parallèle nous avancions la v1.

# Version v1 – projet Graphique seul

## Éléments d’interface

## Implémentation et tests

### 6.1 Implémentation

### Liste et rôle des fichiers de cette version :

|  |  |
| --- | --- |
| lecteurVue.h | Spécification de la classe graphique Qt contenant l’interface du lecteur de diaporamas |
| lecteurVue.cpp | Corps de la classe LecteurVue |
| lecteurvue.ui | Fichier du dessin de l’interface réalisé par QtDesigner |
| main.cpp | Teste les méthodes de la classe Lecteur |

### Remarques sur l’implémentation :

### Commenter brièvement les choix importants d’implémentation réalisés, comme par exemple, les signals/slots

### 6.2 Test

### A faire :

### Décrire les tests prévus / réalisés pour montrer :

### Le comportement de l’interface non lié aux aspects fonctionnels du programme

### Le comportement de l’interface liée aux aspects fonctionnels du programme

### 6.3 Temps passé

### Pour la v1, du fait que nous avancions en parallèle de la v0 nous avons passé aux alentours de deux semaines (en temps écoulé) celui-ci représente environ 3h/4h de

# Version v2 –

## Diagramme de classes (UML)

*A faire – s’il y a des changements - sinon indiquer que idem v0*

## Comportement de l’application

### Diagramme états-transitions-actions du lecteur de diaporamas (v2)

*A faire*

Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v2

### Dictionnaire des états, événements et Actions (v2)

**Dictionnaire** **des états du diaporama**

|  |  |
| --- | --- |
| *nomEtat* | *Signification* |
|  |  |
|  |  |

Tableau 2 : États du lecteur de diaporamas – v2

**Dictionnaire des événements faisant changer le diaporama d’état**

|  |  |
| --- | --- |
| *nomEvénement* | *Signification* |
|  |  |
|  |  |

Tableau 3 : Evénements faisant changer le diaporama d’état – v2

**Description des actions réalisées lors de la traversée des transitions**

|  |  |
| --- | --- |
| *nomAction* | *Signification* |
|  |  | |
|  |  | |

Tableau 4 : Actions à réaliser lors des changements d’état – lecteur de diaporamas v2

### Table T\_EtatsEvenementsActions (v2)

**Correspondance** matricielle du diagramme états-transitions de l’application :

* en *ligne : les* ***états*** du lecteur de diaporamas (éventuel état de départ d’une transition)
* en *colonne : les* ***événements*** faisant changer le lecteur d’état (déclencheur d’une transition)
* dans chaque cellule : l’état d’arrivée de la transition + action/traitement à faire + éventuellement garde accompagnant la transition

|  |  |  |  |
| --- | --- | --- | --- |
| *Élément graphique pregnant en charge cet événement 🡪* |  |  |  |
| *Événement 🡪*  *nomEtat* |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tableau 5 : Matrice d’états-transitions du lecteur de diaporamas – v2

*L’intérêt de cette vue matricielle est qu’elle permet une préparation naturelle et aisée de l’étape suivante de programmation.*

## Implémentation et tests

### Implémentation (v2)

### Liste et rôle des fichiers de cette version :

|  |  |
| --- | --- |
| lecteurVue.h | Spécification de la classe graphique Qt contenant l’interface du lecteur de diaporamasPréciser le rôle |
| lecteurVue.cpp | Corps de la classe LecteurVue. |
| lecteurvue.ui | Fichier du dessin de l’interface réalisé par QtDesigner |
| lecteur.h | Spécification de la classe Lecteur.Préciser le rôle |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Spécification de la classe ImagePréciser le rôle |
| image.cpp | Corps de la classe Image |
| main.cpp | ?? |

### Remarques sur l’implémentation :

### Commenter brièvement les choix importants d’implémentation réalisés, comme par exemple, les signals/slots

### Tests (v2)

### A faire :

### Décrire les tests prévus / réalisés pour montrer :

### Le comportement de l’interface non lié aux aspects fonctionnels du programme

### Le comportement de l’interface liée aux aspects fonctionnels du programme

### Le comportement fonctionnel de l’application

# Version v5 –

## Diagramme de classes (UML)

*A faire – s’il y a des changements - sinon indiquer que idem vXX*

## Comportement de l’application

### Diagramme états-transitions-actions du lecteur de diaporamas (v5)

*A faire*

Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v5

### Dictionnaire des états, événements et Actions (v5)

**Dictionnaire** **des états du diaporama**

|  |  |
| --- | --- |
| *nomEtat* | *Signification* |
|  |  |
|  |  |

Tableau 2 : États du lecteur de diaporamas – v5

**Dictionnaire des événements faisant changer le diaporama d’état**

|  |  |
| --- | --- |
| *nomEvénement* | *Signification* |
|  |  |
|  |  |

Tableau 3 : Evénements faisant changer le diaporama d’état – v5

**Description des actions réalisées lors de la traversée des transitions**

|  |  |
| --- | --- |
| *nomAction* | *Signification* |
|  |  | |
|  |  | |

Tableau 4 : Actions à réaliser lors des changements d’état – lecteur de diaporamas v5

### Table T\_EtatsEvenementsActions (v5)

**Correspondance** matricielle du diagramme états-transitions de l’application :

* en *ligne : les* ***états*** du lecteur de diaporamas (éventuel état de départ d’une transition)
* en *colonne : les* ***événements*** faisant changer le lecteur d’état (déclencheur d’une transition)
* dans chaque cellule : l’état d’arrivée de la transition + action/traitement à faire + éventuellement garde accompagnant la transition

|  |  |  |  |
| --- | --- | --- | --- |
| *Élément graphique pregnant en charge cet événement 🡪* |  |  |  |
| *Événement 🡪*  *nomEtat* |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tableau 5 : Matrice d’états-transitions du lecteur de diaporamas – v5

*L’intérêt de cette vue matricielle est qu’elle permet une préparation naturelle et aisée de l’étape suivante de programmation.*

## Implémentation et tests

### Implémentation (v5)

### Liste et rôle des fichiers de cette version :

|  |  |
| --- | --- |
| lecteurVue.h | Spécification de la classe graphique Qt contenant l’interface du lecteur de diaporamasPréciser le rôle |
| lecteurVue.cpp | Corps de la classe LecteurVue |
| lecteurvue.ui | Fichier du dessin de l’interface réalisé par QtDesigner |
| lecteur.h | Spécification de la classe LecteurPréciser le rôle |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Spécification de la classe ImagePréciser le rôle |
| image.cpp | Corps de la classe Image |
| main.cpp | ?? |

### Remarques sur l’implémentation :

### Commenter brièvement les choix importants d’implémentation réalisés, comme par exemple, les signals/slots

### Tests (v5)

### A faire :

### Décrire les tests prévus / réalisés pour montrer :

### Le comportement de l’interface non lié aux aspects fonctionnels du programme

### Le comportement de l’interface liée aux aspects fonctionnels du programme

### Le comportement fonctionnel de l’application

## Bilan

Dépôt Git où trouver le projet complet (les versions réalisées)

Temps global de travail (pour le groupe)

Apprentissages majeurs

Difficultés majeures

Points positifs / négatifs de l’activité