S2.01 - Développement d’une application

Documentation des classes



BARLIC François

BOURCIEZ Maxime

DUMAI Étienne

TDII - TP3

Table des matières :

[I. Diagramme de classes 3](#_gjdgxs)

[II. Type nécessaires déclarés 4](#_1fob9te)

[Diaporamas 4](#_3znysh7)

[ImagesDuDiaporama 4](#_2et92p0)

[Images 4](#_tyjcwt)

[VII. Classe Lecteur 5](#_3dy6vkm)

[Attributs 5](#_1t3h5sf)

[Méthodes publiques 5](#_4d34og8)

[VIII. Classe Diaporama 6](#_2s8eyo1)

[Attributs 6](#_17dp8vu)

[Méthodes publiques 6](#_3rdcrjn)

[IX. Classe ImageDansDiaporama 7](#_26in1rg)

[Héritage 7](#_lnxbz9)

[Attributs 7](#_35nkun2)

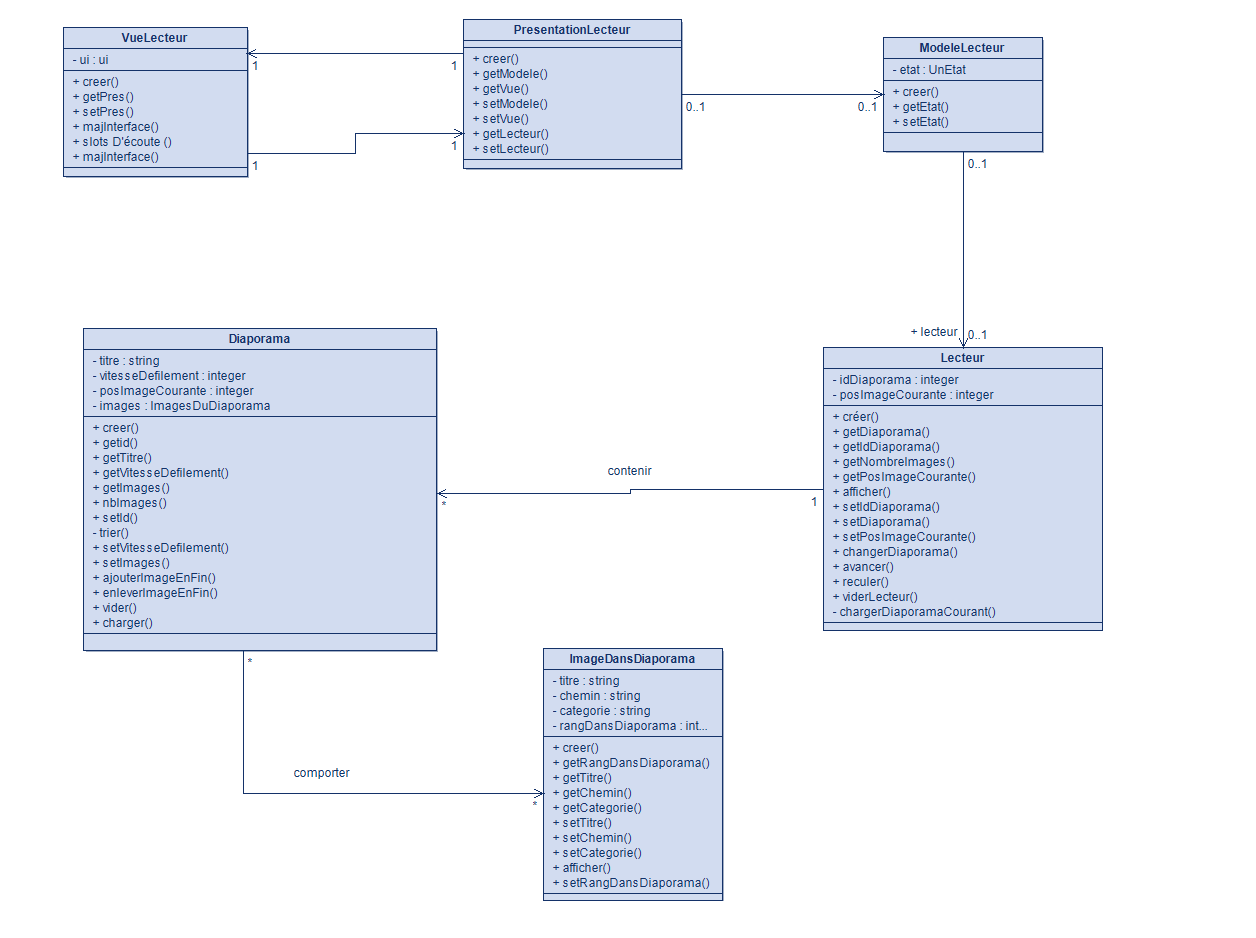
[Méthodes publiques 7](#_1ksv4uv)

[X. Classe Image 8](#_44sinio)

[Attributs 8](#_2jxsxqh)

[Méthodes publiques 8](#_z337ya)

## Diagramme de classes



###### Figure 1 - Diagramme des classes du lecteur de diaporama

Commentaires :

* L’association “contenir” se traduira dans le code par un attribut nommé “allDiapos” contenant tous les diaporamas du lecteur dans la classe Lecteur
* L’association “comprendre” se traduire dans le code par un attribut nommé “localisationImages” contenant les images du diaporama (classe ImageDansDiaporama) dans la classe Diaporama
* Les associations entre les 3 classes du modèle MVP se traduisent par l’ajout des attributs de la classe pointant dans la classe pointée.

## Type nécessaires déclarés

### ImagesDuDiaporama

### **Alias :**

* ImagesDuDiaporama : Alias pour un vecteur d'objets de type ImageDansDiaporama.

### **Description :**

Ce typedef permet de définir un nouveau type ImagesDuDiaporama qui est en réalité un alias pour un vecteur d'objets de type ImageDansDiaporama. Cela facilite l'utilisation de ce type de données et rend le code plus lisible et plus expressif.

### **Utilisation :**

* ImagesDuDiaporama peut être utilisé pour stocker une collection d'images dans un diaporama.

### Images

1. Description : Alias pour un vecteur d'Image.
2. Utilisation : Permet de définir un type Images qui peut être utilisé pour stocker une collection d'images.
3. Déclaration : typedef vector<Image> Images;
4. Exemple d'utilisation : Images mesImages;

## Classe Lecteur

### Attributs privés :

* unsigned idDiaporama; : Identifiant en base de données du diaporama courant. Il est égal à 0 s'il n'y a pas de diaporama dans le lecteur.
* Diaporama\* diaporama; : Pointeur vers le diaporama associé au lecteur. Il est nullptr s'il n'y a pas de diaporama dans le lecteur.
* unsigned int posImageCourante; : Position de l'image courante dans le diaporama courant. Cette valeur est indéfinie lorsque le lecteur est vide ou lorsque le diaporama est vide. Elle est supérieure ou égale à 0 lorsque le lecteur est non vide et que le diaporama est non vide.

### Méthodes publiques :

### Constructeurs et destructeur :

* Lecteur(); : Constructeur de la classe.
* ~Lecteur(); : Destructeur de la classe.

### Getters

* unsigned int getIdDiaporama() const; : Retourne l'identifiant du diaporama associé au lecteur.
* Diaporama\* getDiaporama() const; : Retourne un pointeur vers le diaporama associé au lecteur.
* unsigned int getPosImageCourante() const; : Retourne la position de l'image courante dans le diaporama.
* bool lecteurVide() const; : Retourne vrai si aucun diaporama n'est associé au lecteur, faux sinon.
* ImageDansDiaporama\* getImageCourante() const; : Retourne un pointeur vers l'image courante dans le diaporama.
* unsigned int nbImages() const; : Retourne le nombre d'images dans le diaporama.

### Setters

* void setIdDiaporama(unsigned int pIdDiaporama); : Définit l'identifiant du diaporama associé au lecteur.
* void setDiaporama(Diaporama\* pDiaporama); : Définit le diaporama associé au lecteur.
* void setPosImageCourante(unsigned int pPosImageCourante); : Définit la position de l'image courante dans le diaporama.

### Autres méthodes

* void afficher(); : Affiche les informations sur le lecteur, éventuellement le diaporama et l'image courante.
* void changerDiaporama(unsigned int pId, string pTitre="", unsigned int pVitesse=0); : Permet de choisir un diaporama.
* void avancer(); : Incrémente la position de l'image courante dans le diaporama.
* void reculer(); : Décrémente la position de l'image courante dans le diaporama.
* void viderLecteur(); : Enlève le diaporama courant du lecteur s'il existe.

## Classe Diaporama

### Attributs

* unsigned int id; : Identifiant du diaporama dans la base de données.
* string titre; : Titre du diaporama.
* unsigned int vitesseDefilement; : Vitesse de défilement des images du diaporama.
* ImagesDiaporama images; : Vecteur de pointeurs sur les objets ImageDansDiaporama de ce diaporama.

### Méthodes publiques

#### Constructeur et destructeur

* Diaporama(); : Constructeur de la classe.
* ~Diaporama(); : Destructeur de la classe.

#### Getters

* unsigned int getId() const; : Retourne l'identifiant du diaporama.
* string getTitre() const; : Retourne le titre du diaporama.
* int getVitesseDefilement() const; : Retourne la vitesse de défilement des images du diaporama.
* ImagesDiaporama getImages() const; : Retourne un vecteur de pointeurs sur les objets ImageDansDiaporama du diaporama.
* unsigned int nbImages() const; : Retourne le nombre de pointeurs d'images contenus dans le diaporama.

#### Setters

* void setId(unsigned int pId); : Définit l'identifiant du diaporama.
* void setTitre(string pTitre); : Définit le titre du diaporama.
* void setVitesseDefilement(unsigned int pVitesseDefilement); : Définit la vitesse de défilement des images du diaporama.
* void setImages(const ImagesDiaporama& pImages); : Définit les images du diaporama.

#### Autres méthodes

* void ajouterImageEnFin(ImageDansDiaporama\* pImage); : Ajoute une image à la fin du diaporama.
* void enleverImageEnFin(); : Enlève la dernière image du diaporama et supprime l'objet image enlevé.
* void vider(); : Enlève toutes les images du diaporama et supprime chaque objet enlevé.
* void charger(); : Charge dans le diaporama les images associées au diaporama courant.

### Méthodes privées :

* void trierParRangCroissant(); : Trie les images du diaporama par ordre de rang croissant.

## Classe ImageDansDiaporama

### Attributs

* unsigned int rangDansDiaporama; : Rang de l'image au sein du diaporama auquel l'image est associée.
* string titre; : Intitulé de l'image.
* string categorie; : Catégorie de l'image (personne, animal, objet).
* string chemin; : Chemin complet vers le dossier où se trouve l'image.

### Méthodes publiques

#### Constructeur et destructeur

* ImageDansDiaporama(unsigned int pRangDansDiaporama=0, string pCategorie="", string pTitre="", string pChemin = ""); : Constructeur de la classe.
* ~ImageDansDiaporama(); : Destructeur de la classe.

#### Getters

* unsigned int getRangDansDiaporama() const; : Retourne le rang de l'image dans le diaporama.
* string getCategorie() const; : Retourne la catégorie de l'image.
* string getTitre() const; : Retourne le titre de l'image.
* string getChemin() const; : Retourne le chemin complet vers le dossier où se trouve l'image.

#### Setters

* void setRangDansDiaporama(unsigned int pRangDansDiaporama); : Définit le rang de l'image dans le diaporama.
* void setTitre(string pTitre); : Définit le titre de l'image.
* void setCategorie(string pCategorie); : Définit la catégorie de l'image.
* void setChemin(string pChemin); : Définit le chemin complet vers le dossier où se trouve l'image.

#### Autres méthodes

* void afficher(); : Affiche tous les champs de l'image.

1. Classe LecteurVue

### Attributs privés :

* Ui::LecteurVue \*ui; : Cet attribut semble être une référence à l'interface utilisateur définie par la classe Ui::LecteurVue, probablement générée à partir d'un fichier de conception graphique (comme un fichier .ui dans le cadre de Qt).
* PresentationLecteur\* \_pres; : C'est une référence à un objet de type PresentationLecteur, qui semble être le présentateur associé à cette vue.

### Méthodes publiques :

* LecteurVue(QWidget \*parent = nullptr); : Constructeur de la classe, qui peut éventuellement recevoir un widget parent en paramètre.
* ~LecteurVue(); : Destructeur de la classe.
* PresentationLecteur\* getPres() const; : Méthode pour obtenir une référence vers l'objet PresentationLecteur associé à cette vue.
* void setPres(PresentationLecteur\*); : Méthode pour définir l'objet PresentationLecteur associé à cette vue.
* void majInterface(ModeleLecteur::UnEtat); : Méthode pour mettre à jour l'interface en fonction de l'état donné du modèle Lecteur.

### Slots publics :

* Slots liés aux diaporamas : demanderAvancer(), demanderReculer(), demanderArreterDiapo().
* Slots pour gérer les actions propres au lecteur : demanderChangementDiaporama(), demanderLancementDiapo(), demanderFermetureLecteur(), demanderInformations(), demanderChangementModeAuto(), demanderChangementModeManuel(), demanderChangementVitesseDefilement().
* majInterface(ImageDansDiaporama\*); : Slot pour recevoir et afficher une image dans le diaporama.

1. Classe PresentationLecteur

### Attributs privés :

* LecteurVue\* \_vue; : Une référence à l'objet de la classe LecteurVue, représentant la vue associée à ce présentateur.
* ModeleLecteur\* \_modele; : Une référence à l'objet de la classe ModeleLecteur, représentant le modèle associé à ce présentateur.

### Méthodes publiques :

* PresentationLecteur(); : Constructeur de la classe.
* LecteurVue\* getVue() const; : Méthode pour obtenir une référence vers l'objet LecteurVue associé à ce présentateur.
* ModeleLecteur\* getModele() const; : Méthode pour obtenir une référence vers l'objet ModeleLecteur associé à ce présentateur.
* void setVue(LecteurVue\*); : Méthode pour définir l'objet LecteurVue associé à ce présentateur.
* void setModele(ModeleLecteur\*); : Méthode pour définir l'objet ModeleLecteur associé à ce présentateur.

### Signaux :

* faireAvancer(); : Signal émis pour indiquer que le présentateur doit faire avancer le diaporama.
* faireReculer(); : Signal émis pour indiquer que le présentateur doit faire reculer le diaporama.

### Slots publics :

* Slots pour gérer les actions relatives aux diaporamas : demanderAvancer(), demanderReculer(), demanderArretDiapo(), demanderChangerVitesse().
* Slots pour gérer les actions liées au lecteur : demanderChargement(), demanderLancement(), demanderChangementModeVersManuel(), demanderChangementModeVersAutomatique().

1. Classe ModeleLecteur

### Attributs privés :

* UnEtat \_etat; : Un attribut représentant l'état actuel du modèle, défini comme un type énuméré.
* Lecteur\* \_lecteur; : Une référence à un objet de la classe Lecteur, qui semble être une classe associée à la logique du lecteur.

### Méthodes publiques :

* explicit ModeleLecteur(Lecteur\*, UnEtat = UnEtat::Initial); : Un constructeur qui prend un pointeur vers un objet Lecteur et un état initial en paramètres.
* ModeleLecteur(); : Un constructeur par défaut.
* UnEtat getEtat() const; : Méthode pour obtenir l'état actuel du modèle.
* Lecteur\* getLecteur() const; : Méthode pour obtenir un pointeur vers l'objet Lecteur associé à ce modèle.
* void setEtat(ModeleLecteur::UnEtat); : Méthode pour définir l'état du modèle.
* void setLecteur(Lecteur\*); : Méthode pour définir l'objet Lecteur associé à ce modèle.

### Signaux :

* envoyerInfosMajInterface(ImageDansDiaporama\* image); : Signal émis pour envoyer des informations à la vue pour mettre à jour l'interface avec une image dans le diaporama.

### Slots publics :

* void demandeAvancement(); : Slot pour gérer la demande d'avancement du diaporama.
* void demandeReculement(); : Slot pour gérer la demande de reculement du diaporama.