Saé 2.01

Lecteur de CDs
modélisation UML / Programmation événementielle /
C++ / Qt
éléments de correction

Table des matières

1 Scénarios et Diagrammes de séquence système

1.1 Décrire sous forme textuelle les principaux scénarios d'utilisation (alternatifs) du lecteur.

Exemples de scénarios alternatifs possibles :

Scénario alternatif A1 : Le CD tourne en boucle

Ce scénario remplace le point 9) du scénario nominal.

9a) Une fois la diffusion du titre en cours terminée, le lecteur passe en début de titre suivant et continue la diffusion.
9a1) Une fois la diffusion du dernier titre du cd terminée, le lecteur passe en début du premier titre et continue la diffusion.

Le scénario ne s'arrête jamais s'il n'y a pas d'intervention de l'utilisateur.

Scénario alternatif A2 : L'utilisateur interrompt momentanément un morceau (pause)
avant de le relancer

Ce scénario remplace le point 9) du scénario nominal.

9b) L'utilisateur met la diffusion en Pause.

9b2) Le système arrête la diffusion. Il indique le titre du CD, le nombre total de pistes, le n° de la piste en cours, le titre en cours et le temps de diffusion / temps total du titre
9b3) L'utilisateur relance la lecture

9b4) Le système redémarre la diffusion à partir du point d'arrêt. Il indique le titre du CD, le nombre total de pistes, le n° de la piste en cours, le titre en cours et le temps de diffusion / temps total du titre
9b5) L'utilisateur Stoppe la lecture

Le scénario reprend au point 10 du scénario nominal.

Scénario alternatif A3 : L'utilisateur passe au morceau suivant en cours de lecture

Scénario alternatif A4 : L'utilisateur passe au morceau précédent en cours de lecture

Scénario alternatif A5 : L'utilisateur ouvre le tiroir en cours de lecture, puis re-ferme le tiroir avant de relancer la lecture

1.2 Établir le Diagramme de séquence système associé au scénario nominal fourni en Annexe 1 + 1 scénario alternatif de votre choix.

Les **messages** à destination du système seront bien mis en évidence (par exemple soulignés) de sorte à faciliter l'identification des **méthodes** (préparation pour la question 2.-)

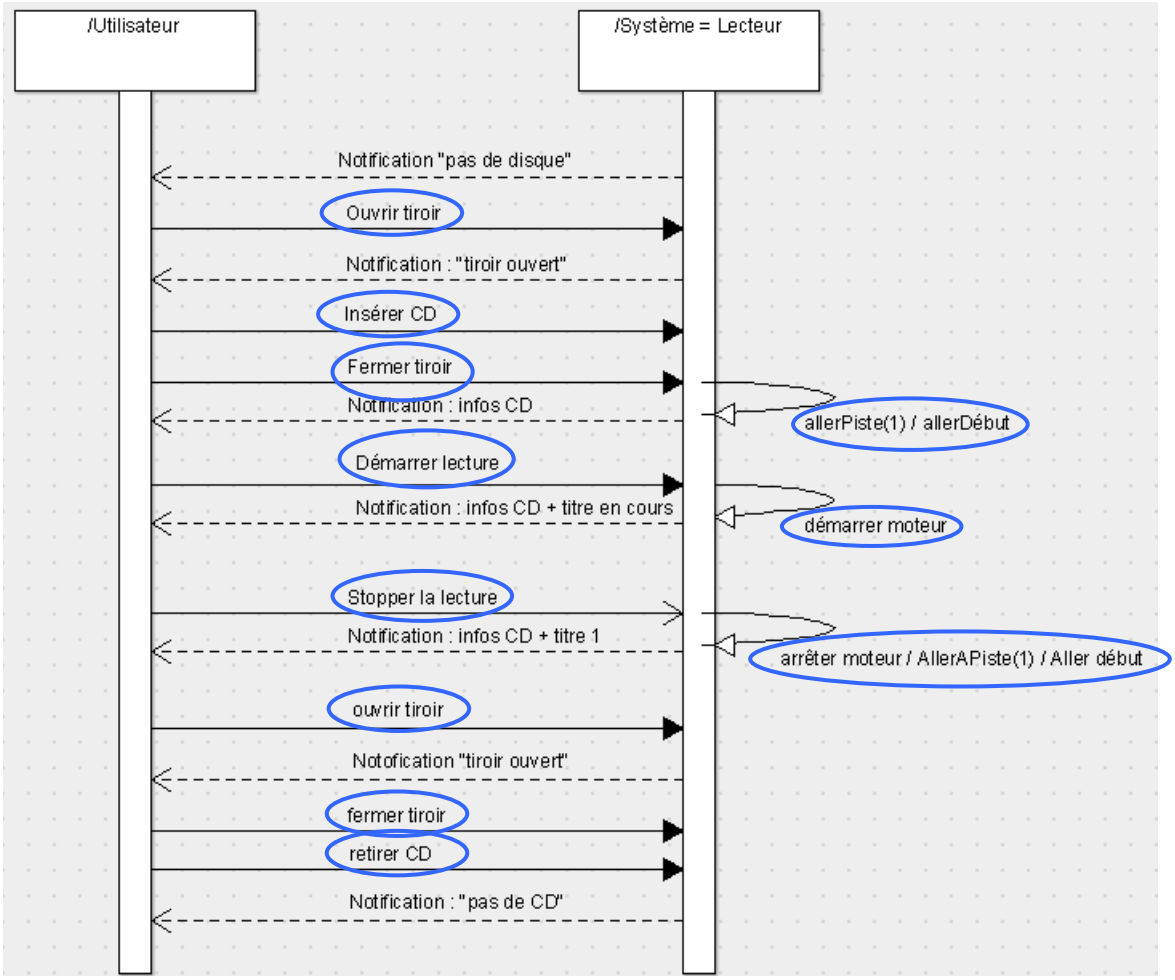


Figure 1 : Diagramme de séquence système d'un scénario

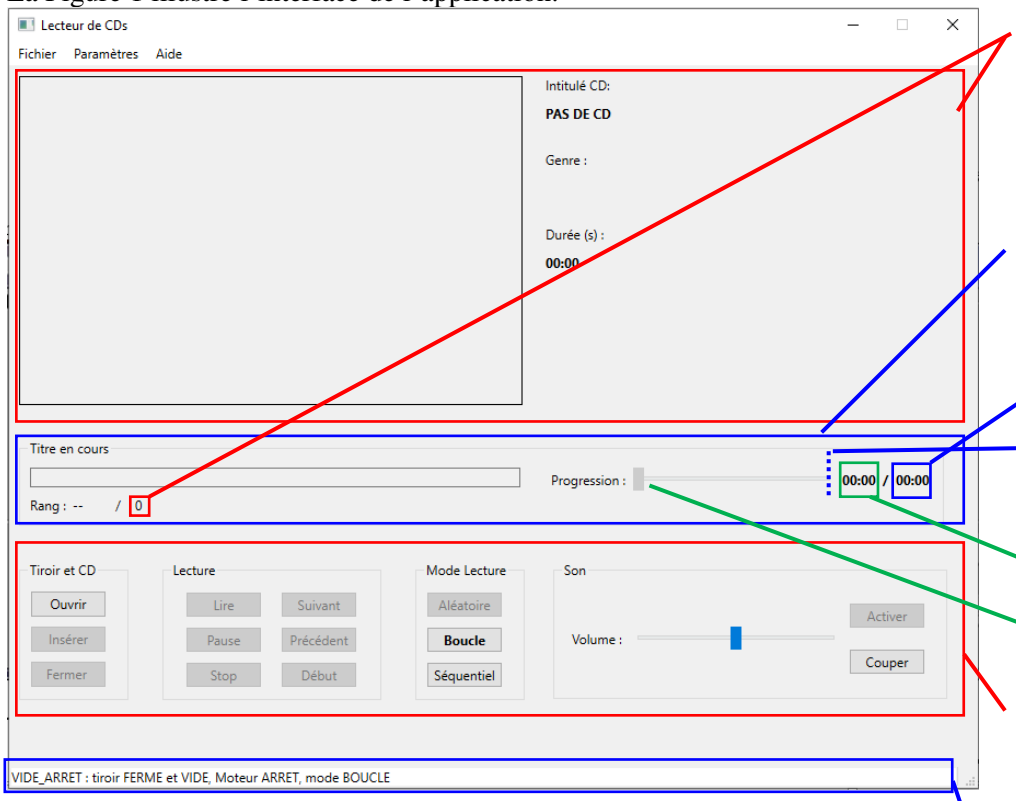
Remarques pour les enseignants :

- L'avantage du diagramme de séquence est la mise en évidence obligée des messages de l'utilisateur ➔ système, qui sont des candidats à devenir des méthodes de la classe Lecteur (ou de ses composants). Ces 'futurs' noms de méthodes se voient moins dans le texte des scénarios textuels. Il faut donc trouver un moyen de les mettre en évidence sans pour autant passer par l'élaboration de tous les diagrammes de séquence, sinon, on ne terminera jamais l'étude dans le temps imparti…
- À noter la forme des 'retours' du système vers l'utilisateur. Ce sont des flèches pointillées pour signifier qu'il ne s'agit pas de messages qui seront susceptibles de se transformer en méthodes… de l'Utilisateur ! Il s'agit par contre bien de ***retours d'informations*** vers l'utilisateur, qui se traduiront probablement par de l'affichage d'information s ou d'un retour sonore…

2 Interface de l'application

2.1 Interface

La Figure 1 illustre l'interface de l'application.



Zone infos du CD :
- intitulé,
- genre,
- durée totale,
- pochette,
- nbre total de titres

Zone infos sur titre en cours :
- intitulé,
- durée su titre, sous 2 formats :
…valeur numérique en secondes
…longueur max. de la barre de défilemment
- ***progression de la lecture sous 2 formats*** :
…durée de diffusion en secondes
…position du curseur sur la

Zone de commande du lecteur :
- commande du tiroir,
- commande de lecture,
- mode de lecture,
- réglage du son

Barre d'état

Figure 2 : Interface du lecteur de CDs

2.2 Remarques

L'interface comprend 4 zones principales :

- Zone infos du CD : Elle se met à jour lorsque l'utilisateur insère / retire un CD
- Zone infos sur le titre en cours : Elle comprend 2 types d'informations :
  - Informations sur le titre courant qui se mettent à jour à chaque changement de titre courant
  - Progression de la lecture du titre courant = temps de diffusion du titre. Elle est mise à jour durant la lecture du titre courant. Elle est exprimée sous 2 formes : via un nombre exprimé en secondes, et via le déplacement du curseur sur la barre de Progression, dont la longueur totale représente la durée totale de diffusion du titre.
- Zone des commandes de manipulation du lecteur CD. Elle contient les widgets permettant de commander le lecteur CD.
- Barre d'état, qui détaille l'état du lecteur de CDs et l'état de chacun de ses composants.

- Quelques règles ergonomiques à respecter lors de la conception d'une interface :
  – Les éléments relevant d'une même signification doivent être regroupés. Exemples : groupement Tiroir & CD, Lecture, Mode lecture, Progression, …
  – Les éléments graphiques doivent être alignés : à gauche, en haut, en bas, …

  Le respect de ces règles minimales st plus important que la recherche de pictogrammes !

3 Diagramme de classes de l'application et description des classes

3.1 Établir le diagramme de classes UML de l'application

Consigne : On se focalisera uniquement sur les **classes métier**, cad celles décrivant le lecteur et ses composants indépendamment des éléments d'interface que comportera le simulateur. Cela correspond à modéliser la partie **Modèle** dans la terminologie MVP.

Le schéma est décrit dans la Figure 3.

3.2 Remarques – Explications

Selon les informations fournies dans les spécifications :
Un lecteur est composé d'une **Cellule**, d'un **TiroirCD** et d'une **SortieSon**

- La Cellule est chargée de lancer le moteur qui fera tourner le CD et contient également le dispositif (tête de lecture) qui lit chaque titre du CD (). La position de la tête de lecture sur le titre permet de savoir le temps de diffusion écoulé, et de reconnaître la fin du titre.
- Le TiroirCD est le dispositif permettant de recueillir physiquement le CD dans le lecteur. Par contre, le lecteur ne reconnaît la présence du CD que lorsque le tiroir est fermé.
- Association entre CD et Tiroir : le CD ne fait pas partie du lecteur. Il est déposé dans le tiroir afin d'être lu. La modélisation aurait aussi pu associer le CD au lecteur, pour représenter le fait que le CD est ***lu*** par le lecteur.
- Un CD est composé de un ou plusieurs titres.
- **Concernant les attributs encadrés en rouge : `titreEnCours` et `laVue`**
  Ils ne font pas partie de la vision conceptuelle **métier** demandée par la Consigne, et donc à ce titre, ils n'auraient pas dû figurer dans ce schéma initial. Nous les avons toutefois laissés afin de simplifier le document d'analyse et vous donner un nombre limité de versions du shéma.
  - Le pointeur **`laVue`** implémente le lien vers l'interface, contrairement à l'objectif de ne modéliser que le système d'information, le Modèle sous-jacent de l'application.
  - Le pointeur **`titreEnCours`** dont la présence est un *choix d'implémentation* apparaissant plus tard, lors de la mise en œuvre du modèle. En effet, lors de l'implémentation du modèle, le lecteur aura très souvent besoin d'utiliser les informations liées au titre en cours de diffusion. Plusieurs choix étaient possibles. Nous avons choisi celle où le lecteur garde une référence vers le titre en cours de diffusion.
- **Concernant les classes encadrées en rouge :**
  Même remarque que pour els attributs. Elles ont liées à l'implémentation du lecteur avec Qt et non pas de la vision conceptuelle **métier** demandée par la Consigne.

3.3 Remarques méthodologiques

- 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.
- Afin d'alléger le schéma et d'en faciliter la lecture, les destructeurs ont été omis, ainsi que les getters et setters des classes Cd, Titre et LecteurCD.
- Dasns toutes les classes : la démarche objet veut que :
  - Dès qu'il y a un attribut, il faut 1 méthode pour le modifier et une autre pour le consulter. Les méthodes de consultation ont été appelées getXX. Les méthodes de modification (setXX) ont gardé les noms des opérations courantes qui allaient être appliquées aux objets (ouvrirTiroir(), fermerTiroir(), arrêterMoteur(), etc…)

- Inversement, dès que l'on identifie une méthode qui fait changer un objet d'état (ouvrirTiroir(), arrêterMoteur(), …), il faut un attribut qui mémorise cet état).
- En appliquant ces 2 principes, on arrive à trouver les méthodes/attributs sans forcément connaître à fond le domaine d'application du schéma de classes

3.4 Description des classes : dictionnaires des éléments et des méthodes :

Voir Annexes.

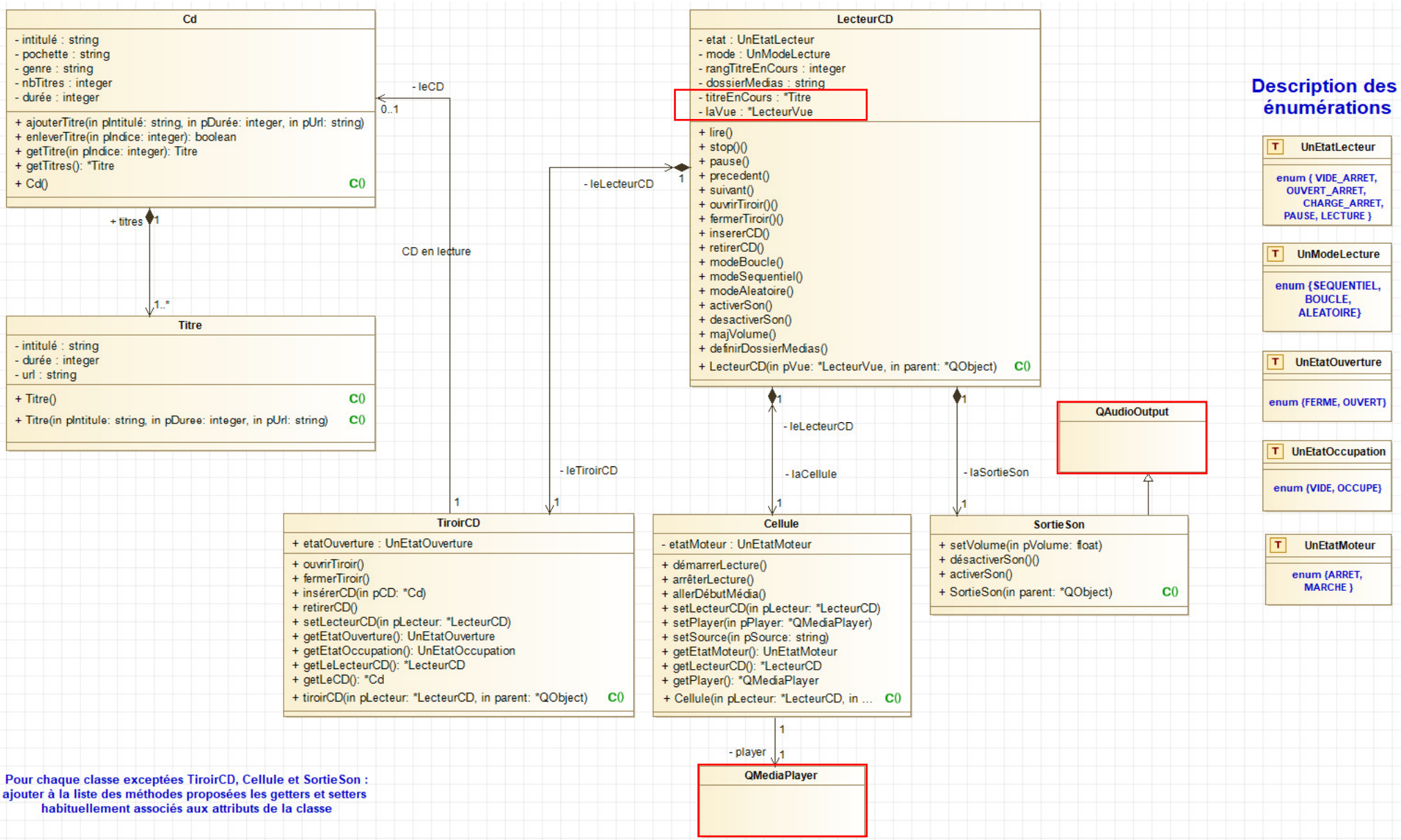


Figure 3 : Diagramme de classes de l'application (formalisme UML)

4 Diagrammes états-transitions du lecteur de CD (classe LecteurCD)

4.1 États du lecteur en fonction des états de ses composants

	Composant / *nomEtatLecteur*	TiroirCD		Cellule		
		étatTiroir	occupation Tiroir	étatMoteur	rangTitreEn Cours	Temps Diffusion
		~~OUVERT~~	~~OCCUPE~~	~~MARCHE~~	--	--
1	**OUVERT_ARRET**	OUVERT	OCCUPE	ARRET	--	--
		~~OUVERT~~	~~VIDE~~	~~MARCHE~~	--	--
1	**OUVERT_ARRET**	OUVERT	VIDE	ARRET	--	--
2	**LECTURE**	FERME	OCCUPE	MARCHE	>= 1	>= 0
3	**PAUSE**	FERME	OCCUPE	ARRET	>= 1	> 0
4	**CHARGE_ARRET**	FERME	OCCUPE	ARRET	= 1	= 0
		~~FERME~~	~~VIDE~~	~~MARCHE~~	--	--
5	**VIDE_ARRET**	FERME	VIDE	ARRET	--	--

Tableau 1 : Etats du LecteurCd en fonction des états de ses composants

4.2 Remarques

- On a appliqué la méthode préconisée : lister toutes les combinaisons possibles des états des composants du lecteur, puis rayer les cas impossibles, puis donner un nom 'parlant' aux états possibles restants.
- On a regroupé dans un même état (1) deux lignes différentes : en effet, il n'est pas précisé que le lecteur distingue la présence du CD lorsque le tiroir est ouvert. Dans notre cas, le lecteur ne distingue la présence du CD qu'une fois le tiroir fermé. ➔ Cette précision doit figurer dans le paragraphe consacré à l'explicitation des spécifications externes et dont l'objectif est de lever toutes les ambiguïtés.
- Lorsqu'il n'y a pas de valeur dans une case (« --»), cela veut dire que la valeur de cet attribut n'est pas pertinente pour la détermination de l'état du lecteur.
- Etat **CHARGE_ARRET** : c'est le mode **STOP** indiqué dans les spécifications : la tête de lecture se positionne au début du premier titre.
- Le mode de lecture (**BOUCLE**, **SEQUENTIEL** ou **ALEATOIRE**) n'influe pas sur les états du lecteur, de même que l'état de la **SortieSon** (son activé ou pas).

4.3 Diagramme états-transitions du lecteur – formalisme UML – cf. Figure 4

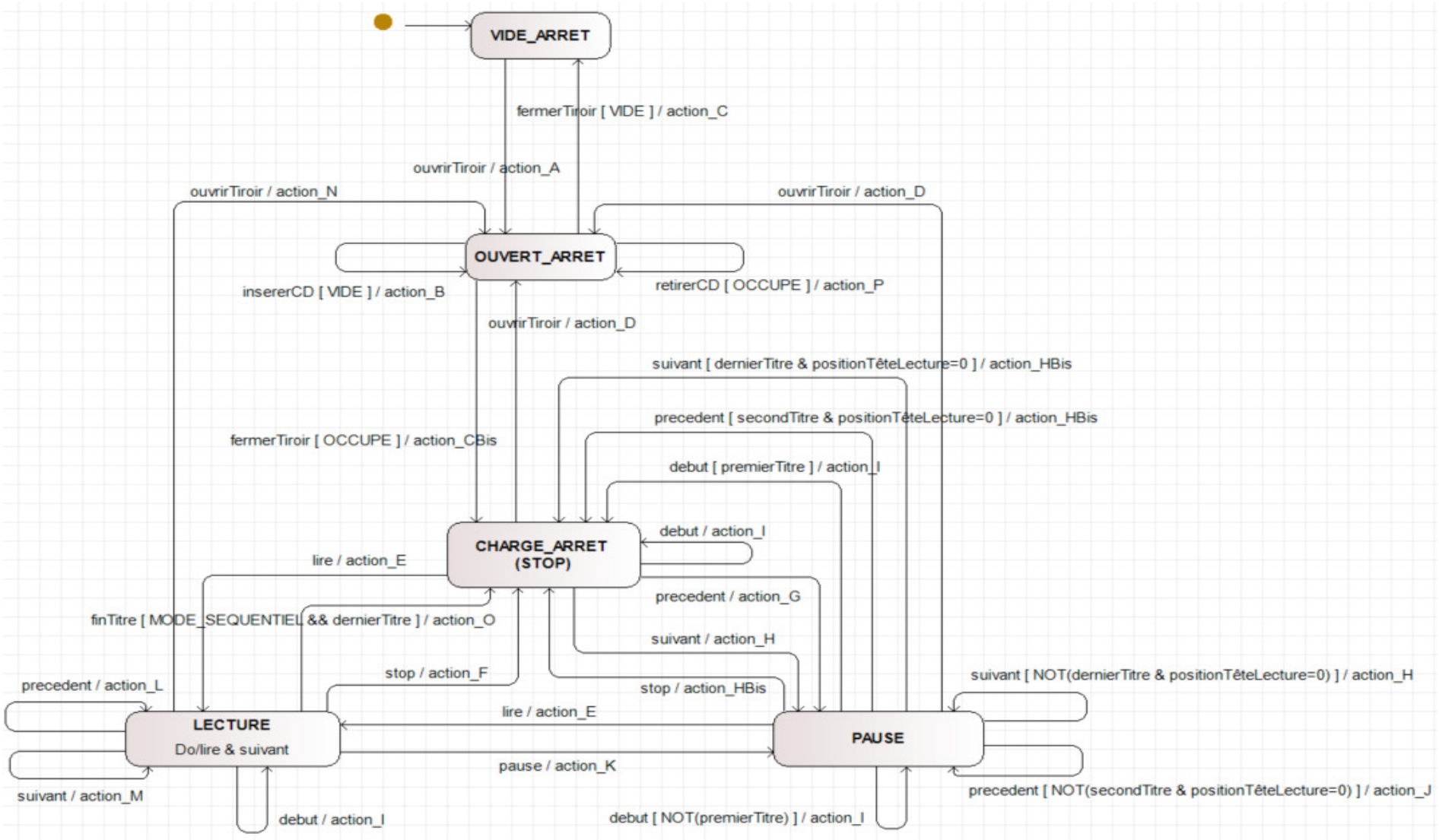


Figure 4 : Diagramme états-transitions du lecteur de CDs – selon le formalisme UML

4.3.1 Remarques concernant le diagramme états-transitions UML

- Le programme peut être arrêté par l'utilisateur à n'importe quel moment (fermeture de la fenêtre d'exécution du programme) ➔ il n'y a pas d'état terminal (tout état peut être un état terminal).
- Les noms des événements sont en correspondance avec les noms des méthodes qui les déclenchent.
  Exemples :
  - l'événement **`stop`** a été généré suite à l'appel de la méthode **`stop()`** effectuée sur le lecteur
  - l'événement **`ouvrirTiroir`** a été généré suite à l'appel de la méthode **`ouvrirTiroir()`** effectuée sur le tiroir.
- Une certaine complexité du schéma vient du fait de la séparation des états **`ARRET_CHARGE (=STOP)`** (rangTitreEnCours=0, tempsDiffusion=0) et **`PAUSE`** (toute autre combinaison de ces 2 attributs).

  Ainsi, lorsque les événements **`précédent`**, **`suivant`**, **`pause`**… seront déclenchés, il faudra toujours vérifier dans quelle situation on est par rapport à tempsDiffusion et rangTitreEnCours afin de basculer sur le bon état (**`ARRET_CHARGE`** ((=**`STOP`**) ou **`PAUSE`**).
- Une activité se déroule à l'intérieur de l'état **`LECTURE`**. En effet, cette action **`lire&Suivant`** (qui dure, c'est pourquoi c'est une activité) consiste pour le lecteur en mode **`SEQUENTIEL`** ou **`BOUCLE`**, à diffuser le morceau courant, puis, une fois le morceau terminé, à passer au morceau (piste) suivant et à le diffuser…. Et tout cela en continu selon les caractéristiques du lecteur fournies.

  Le seul événement faisant sortir le lecteur de l'état **`LECTURE`** est l'événement **`finTitre`** avec la garde **`[MODE_SEQUENTIEL && dernierTitre]`** qui signifie :
  - événement **`finTitre`** : La tête de lecture a atteint la fin du média correspondant au titre courant
  - garde **`[MODE_SEQUENTIEL && dernierTitre]`** : le lecteur est en mode de lecture **`MODE_SEQUENTIEL`** et le titre courant est le dernier titre du CD.

  Dans ce cas, la lecture s'arrête et le lecteur se met en état de lecture **`ARRET_CHARGE (=STOP)`**.

  Dans tous les autres cas, la fin de la lecture du titre courant entraîne le passage au titre suivant et sa lecture, et, une fois le dernier titre du CD lu, le passage au premier titre et sa lecture, pour continuer sans fin.
- Le lecteur dispose d'un autre attribut d'état : l'attribut **`mode`**, de type **`UnModeLecture`** et dont les valeurs sont : **`SEQUENTIEL, BOUCLE, ALEATOIRE`**. Le comportement du lecteur qui a été modélisé ne tient compte que des modes **`SEQUENTIEL, BOUCLE`** et ignore le mode **`ALEATOIRE`**. Dans cette situation, le mode de lecture n'influe en rien sur les états du lecteur, *excepté lors de l'événement* ***`finTitre`*** *décrit dans l'alinéa précédent.* Cela veut dire que l'algorithme/code du gestionnaire d'événement prenant en charge l'événement ***`finTitre`*** devra être structuré en respectant le modèle désormais 'classique' suivant : `switch(getMode() {case SEQUENTIEL : xxx ; break ; case BOUCLE : xxx ; break ; case ALEATOIRE : break}`)

4.3.2 Remarques méthodologiques

- **Lien événement-méthode** : Il est important de comprendre le lien entre ces 2 notions : c'est l'appel d'une méthode du schéma de classes qui déclenche l'événement correspondant.
- Lorsque l'on établit un diagramme états-transitions, bien vérifier que tout événement défini dans ce schéma correspond bien à une méthode du schéma de classes.
- **L'activité Do/lire & suivant** :
  Elle dure tout le temps que le système est dans l'état **`LECTURE`**. : cad qu'elle est sans fin à l'intérieur de l'état **`LECTURE`** et sera interrompue par un des événements faisant sortir le lecteur de cet état **`LECTURE`**.

4.4 Diagramme états-transitions du lecteur – version matricielle – cf. Tableau 2

événements ➔ / *états lecteur* ⬇	**ouvrir Tiroir**	**fermer Tiroir**	**insérer CD**	**retirer CD**	**finTitre**
VIDE_ARRET	**OUVERT_ARRET** / action_A	--	--	--	--
OUVERT_ARRET	--	[VIDE] **VIDE_ARRET** / action_C --- [OCCUPE] **CHARGE_ARRET** / action_CBis	[VIDE] **OUVERT_ARRET** / action_B	[OCCUPE] **OUVERT_ARRET** / action_P	--
LECTURE	**OUVERT_ARRET** / action_N	--	--	--	[MODE_SEQUENTIEL && dernierTitre] **CHARGE_ARRET** / action_O
CHARGE_ARRET (=STOP)	**OUVERT_ARRET** / action_D	--	--	--	--
PAUSE	**OUVERT_ARRET** / action_D	--	--	--	--

événements ➔ / *états lecteur* ⬇	**lire**	**pause**	**stop**	**précédent**	**suivant**	**début**
VIDE_ARRET	--	--	--	--	--	--
OUVERT_ARRET	--	--	--	--	--	--
LECTURE	--	**PAUSE** / action_K	**CHARGE_ARRET** / action_F	**PAUSE** / action_L	**PAUSE** / action_M	**LECTURE** / action_I
CHARGE_ARRET (=STOP)	**OUVERT_ARRET** / action_E	--	--	**PAUSE** / action_G	**PAUSE** / action_H	**CHARGE_ARRET** / action_I
PAUSE	**OUVERT_ARRET** / action_E	--	**CHARGE_ARRET** / action_HBis	[NOT(secondTitre && positionTêteLecture=0)] **PAUSE** / action_J --- [secondTitre && positionTêeteLecture=0] **CHARGE_ARRET** / action_HBis	[NOT(dernierTitre && positionTêteLecture=0)] **PAUSE** / action_H --- [dernierTitre && positionTêeteLecture=0] **CHARGE_ARRET** / action_HBis	[NOT(premierTitre)] **PAUSE** / action_I --- [premierTitre] **CHARGE_ARRET** / action_I

Tableau 2 : Version matricielle du Diagramme Etats/Transitions du Lecteur de CDs

4.5 Dictionnaires associés aux diagrammes états-transitions

4.5.1 Dictionnaire des états du lecteur

nomEtatLecteur	*Signification*
`VIDE_ARRET`	C'est l'état (initial) du lecteur. Pas de disque à l'intérieur et tiroir fermé.
`OUVERT_ARRET`	Le tiroir est ouvert, le lecteur ne communique rien sur la présence ou pas de CD à l'intérieur.
`LECTURE`	Le lecteur est en cours de diffusion d'un titre.
`CHARGE_ARRET`	Le tiroir est fermé et un CD y est présent, le moteur de la cellule est arrêté. La cellule est positionnée en début de piste 1 (et temps diffusion = 0).
`PAUSE`	Le tiroir est fermé et un CD y est présent, le moteur de la cellule est arrêté. La cellule est positionnée sur un titre, et une diffusion a démarré (la position de la tête de lecture dans le titre est > 0). La relance de la lecture du morceau redémarrera au point d'arrêt.

Tableau 3 : Dictionnaire des états du lecteurCD

4.5.2 Dictionnaire des événements faisant changer le lecteur d'état

nomEvénement	*Signification*
`ouvrirTiroir`	Le tiroir du lecteur a été ouvert par l'utilisateur (via un widget de l'interface)
`fermerTiroir`	Le tiroir du lecteur a été fermé par l'utilisateur
`insérerCD`	Un CD a été inséré dans le tiroir du lecteur
`retirerCD`	Le CD présent dans le tiroir du lecteur a été retiré
`lire`	L'utilisateur a demandé la diffusion du titre courant
`pause`	L'utilisateur a arrêté la diffusion du titre courant. Le moteur de la cellule s'arrête. La tête de lecture de la cellule ne bouge pas.
`stop`	L'utilisateur a arrêté la diffusion du titre courant. Le moteur de la cellule s'arrête. La tête de lecture de la cellule est déplacée en début du premier titre.
`précédent`	La tête de lecture de la cellule a été déplacée en début de titre précédent le titre courant. L'état du lecteur n'est pas modifié, excepté si la tête de lecture était en position 0 du second titre du CD.
`suivant`	La tête de lecture de la cellule a été déplacée en début de titre suivant le titre courant. L'état du lecteur n'est pas modifié, excepté si la tête de lecture était en position 0 du dernier titre du CD.
`début`	La tête de lecture de la cellule a été déplacée en début du titre courant. L'état du lecteur n'est pas modifié, excepté si le titrer courant était le premier titre du CD.

Tableau 4 : Dictionnaire des événements faisant changer d'état le lecteur

4.5.3 Dictionnaire des gardes du schéma

Les gardes représentent des conditions sur **des éléments présents** dans le système d'information de l'application. Il faut donc s'assurer que ces informations soient présentes dans le diagramme de classes.

Nom garde	*Signification*
`OCCUPE`	C'est un des états de chargement du tiroir indiquant qu'il contient un CD. C'est une valeur du type énuméré **`UnEtatTiroir`**. Il est accessible via la méthode **`getEtatOccupation()`**.
`VIDE`	C'est un des états de chargement du tiroir indiquant qu'il ne contient pas de CD. C'est une valeur du type énuméré **`UnEtatTiroir`**. Il est accessible via la méthode **`getEtatOccupation()`**.
`secondTitre`, `premierTitre`, `dernierTitre`	Il s'agit du rang du titre en cours de diffusion. Il est accessible via la méthode **`getRangTitreEnCourant()`**
`positionTêteDeLecture`	Elle correspond au temps de diffusion du titre courant depuis son début (en secondes). → Il faudra ajouter cette méthode dans la classe Cellule !

Tableau 5 : Dictionnaire des gardes du diagramme états-transitions du lecteurCD

4.5.4 Dictionnaire des actions du diagramme états-transitions

Nom action sur le diagramme E/T	**Actions métier à réaliser** = Ordres à destination du Modèle	**Actions de changement d'état du lecteur** = ordre à destination de la Présentation	**Actions sur interface** = Ordres à destination de la Vue
`action_A`	- ouvrir le tiroir (il devient **`OUVERT`**)	- changer d'état le lecteur (**`OUVERT_CHARGE`**)	- zone infos CD - zone contrôle du lecteur - barre d'état
`action_B`	- créer un objet CD - le peupler avec les titres prédéfinis - demander au tiroir de charger ce CD (le tiroir devient **`OCCUPE`**)	--	- zone infos CD - zone contrôle du lecteur - barre d'état
`action_P`	- demander au tiroir de retirer ce CD (le tiroir : o détruira l'objet CD et ses titres o dissociera le CD en cours du tiroir (le tiroir devient **`VIDE`**))	--	- zone infos CD - zone contrôle du lecteur - barre d'état
`action_C`	- fermer le tiroir (il devient **`FERME`**)	- changer d'état le lecteur (**`VIDE_ARRET`**)	- zone infos CD (intitulé, pochette, genre, durée totale, nombre de titres) - zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
`action_CBis`	- fermer le tiroir (il devient **`FERME`**) - associer le CD au tiroir (il devient **`OCCUPE`**) - initialiser le titre en cours et son rang avec le premier titre du CD - associer le titre en cours comme source média de la Cellule - positionner la tête de lecture en début de média	- changer d'état le lecteur (**`CHARGE_ARRET`**)	- zone infos CD (intitulé, pochette, genre, durée totale, nombre de titres) - zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
`action_D`	- ouvrir le tiroir (il devient **`OUVERT`**) - le titre en cours du lecteur et son rang deviennent indéfinis - la Cellule n'a plus de plus de source média associée	- changer l'état du lecteur (**`OUVERT_ARRET`**)	- zone infos CD - zone infos Titre - zone contrôle du lecteur - barre d'état

action_E	- démarrer la lecture du titre en cours par la Cellule (le moteur de la Cellule devient ***MARCHE***)	- changer l'état du lecteur (**LECTURE**)	- zone contrôle du lecteur - barre d'état - mise à jour automatique de la zone infos Titre (***position*** et ***progression***) pilotée par la tête de lecture en fonction de l'avancement de la tête de lecture
action_F	- arrêter la lecture du titre en cours par la Cellule (il devient ***ARRET***) - puis action_HBis : - le premier titre du CD (et son rang) deviennent le titre courant (et rang courant) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média	- changer l'état du lecteur (**CHARGE_ARRET**)	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_HBis		- changer l'état du lecteur (**CHARGE_ARRET**)	
action_I	- positionner la tête de lecture en début du média du titre courant	--	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_G	- le dernier titre du CD (et son rang) deviennent le titre courant (et rang courant) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média	- changer l'état du lecteur (**PAUSE**)	- mise à jour automatique de la zone infos Titre (***position*** et ***progression***) pilotée par la Cellule en fonction de la position et avancement de la tête de lecture - zone contrôle du lecteur - barre d'état
action_H	- le titre suivant (et son rang) du titre en cours deviennent le titre en cours (et rang en cours) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média	- changer l'état du lecteur (**PAUSE**)	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_I		--	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état

action_I		- changer l'état du lecteur (**CHARGE_ARRET**)	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_H		--	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_HBis	- le premier titre du CD (et son rang) deviennent le titre courant (et rang courant) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média	- changer l'état du lecteur (**CHARGE_ARRET**)	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_HBis		- changer l'état du lecteur (**CHARGE_ARRET**)	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_J	- le titre précédent (et son rang) du titre en cours deviennent le titre en cours (et rang en cours) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média	--	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - zone contrôle du lecteur - barre d'état
action_K	- arrêter le moteur (le moteur de la Cellule devient **ARRET**)	- changer l'état du lecteur (**PAUSE**)	- zone contrôle du lecteur - barre d'état
action_L	- le titre précédent (et son rang) ddu titre en cours deviennent le titre en cours (et rang en cours) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média - démarrer la lecture du titre courant par la Cellule (le moteur de la Cellule devient ***MARCHE***)	--	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - barre d'état
action_M	- le titre suivant (et son rang) du titre en cours deviennent le titre en cours (et rang en cours) - charger dans la Cellule le média associé et au titre courant - positionner la tête de lecture au début du média - démarrer la lecture du titre courant par la Cellule (le moteur de la Cellule devient ***MARCHE***)	--	- zone infos Titre (intitulé, rang, durée, position=0, progression=0) - barre d'état
action_I		--	

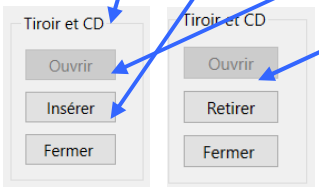
`action_O`	stop() (= invoquer la méthode stop() du lecteurCD)	La méthode stop() se chargera de faire les changements d'état	La méthode s'occupera des mises à jours
`action_N`	- arrêter le moteur de la Cellule(il devient **`ARRET`**) (puis action_D) - ouvrir le tiroir (il devient **`OUVERT`**) - le titre en cours du lecteur et son rang deviennent indéfinis - la Cellule n'a plus de plus de source média associée	- changer l'état du lecteur (**`OUVERT_ARRET`**)	
`action_D`		- changer l'état du lecteur (**`OUVERT_ARRET`**)	
`action_E`		- changer l'état du lecteur (**`LECTURE`**)	

Tableau 6 : Dictionnaire des gardes du diagramme états-transitions du lecteurCD

4.6 Evénements du diagramme états-transitions et éléments d'interface

4.6.1 Tableau de prise en charge des événements par les éléments d'interface

événements →	`ouvrir Tiroir`	`fermer Tiroir`	`insérer CD`	`retirer CD`
→ *Eléments de l'interface prenant en charge cet événement*	`btnOuvrir`	`btnFermer`	`btnInserer Retirer`	`btnInserer Retirer`



→ *Eléments de l'interface prenant en charge cet événement*	`btnLecture`	`btnPause`	`btnStop`	`btnPrec`	`btnSvt`	`btnDeb`
événements →	`lire`	`pause`	`stop`	`précédent`	`suivant`	`début`

Tableau 7 : Liste des événements du lecteur générés par des éléments graphiques (de l'interface)

4.6.2 Remarques

- Pour simplifier l'interface, de manière générale, 1 bouton de l'interface ne prend en charge qu'un seul événement de l'application
- Une exception a été faite, à titre d'exemple : le bouton **`btnInsererRetirer`**, qui génère 2 évènements
  - Evénement **`insérerCD`**
  - Evénement **`retirerCD`**

Ce sera l'occasion de voir les 2 mises en œuvre.

4.7 États du lecteur et états de l'interface

Pour chaque état du lecteur, il faut définir de manière précise l'état de l'interface, c'est-à-dire l'état (visuel) de chaque élément de l'interface.

La démonstration sera faite à titre d'exemple pour un seul état : l'état `VIDE_ARRET`

Elément d'interface	**Description** (visible / invisible ; activé / inactif ; focus ; …)
`btnOuvrir`	Visible, actif
`btnFermer`	Visible, inactif
`btnInsererRetirer`	Visible, inactif, libellé « Insérer »
`btnLecture`	Visible, inactif
`btnPause`	Visible, inactif
`btnPrec`	Visible, inactif
`btnSvt`	Visible, inactif
`btnDeb`	Visible, inactif
`btnAleatoire`	Visible, inactif
`btnBoucle`	Visible, actif, libellé en gras (modeLecture courant)
`btnSequentiel`	Visible, actif
`btnSonActif`	Visible, inactif
`btnSonInactif`	Visible, actif
`sliderVolume`	Visible, actif, placé au milieu de la plage de volumes autorisés
`labelValeurInitituleCD`	PAS DE CD
`labelValeurGenreCD`	-- (vide)
`labelValeurDureeCD`	00:00
`labelValeurTitreEnCours`	-- (vide)
`labelValeurRang`	-- (vide)
`labelValeurNbTitresDansCD`	0
`labelTemps`	00:00
`labelDuree`	00:00
`sliderProgression`	Visible, inactif, curseur place en position 0 – à l'extrémité gauche

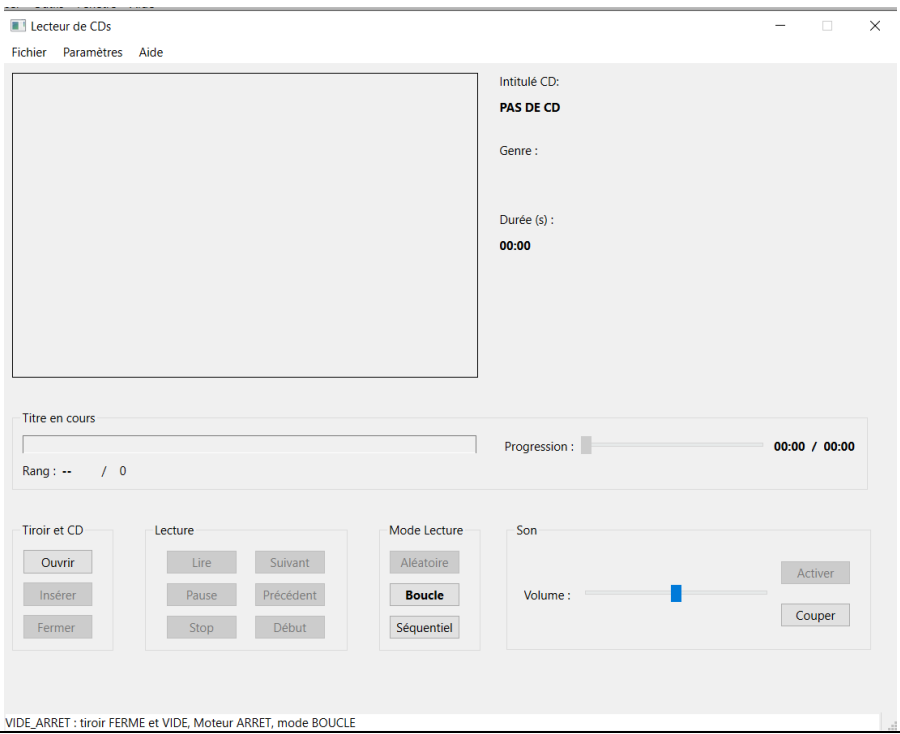


Tableau 8 : Correspondance entre interface et états du lecteur

5 Spécifications internes du programme réalisé

5.1 Organisation du code

Le code est organisé en de la manière suivante :
- Une classe graphique, LecteurVue : contient l'interface de l'application
- La classe LecteurCD représente l'objet composé : le lecteur de CDs.
- Le lecteur de CD est composé de composants : ce sont les classes TiroirCD, Cellule, SortieSon
- Les classes CD et Titre représentent le CD lu par le lecteur

Le code est structuré en suivant le découpage MVP de manière partielle :
- La classe LecteurVue joue le rôle de la **Vue**
- Les classes CD, Titre, Cellule, Tiroir, SortieSon et LecteurCD font partie du **Modèle**
- La classe LecteurCD joue aussi le rôle de la **Présentation**. A ce titre :
    - La classe LecteurCD dispose d'un pointeur sur le Vue pour lui transmettre tous les ordres d'affichage
    - La classe LecteurVue pointe vers le LecteurCD pour lui transmettre toutes les interactions de l'utilisateur. La vue ne communique qu'avec l'objet de la classe LecteurCD.
    - La classe LecteurCD implémente le diagramme états-transition de l'application lecteur de Cds. L'objet lecteur analyse les interactions de l'utilisateur, réalise les répponses en donnat les ordre nécessaires à ses composants, puis envoie à la vue les informations à afficher.

5.2 Ressources fournies

- LecteurCD.h
- CD.h
- Titre.h et Titre.cpp
- TiroirCD.h et TiroirCD.cpp
- SortieSon.h et SortieSon.cpp
- Cellule.h
- méthode void LecteurCD::peuplerCD(Cd *pCD)
- quelques ressources médias pour une version de l'application sans Base de Données.