EISTI 2008-2009 – TD/TP de MDA – Séance 1

Modélisation – Génération de code – Rétro-ingénierie avec NetBeans

0 : Installation de NetBeans 6.5 et du plugin UML :

Téléchargez et installez la version 6.5 de NetBeans pour Java SE (la plus petite) :

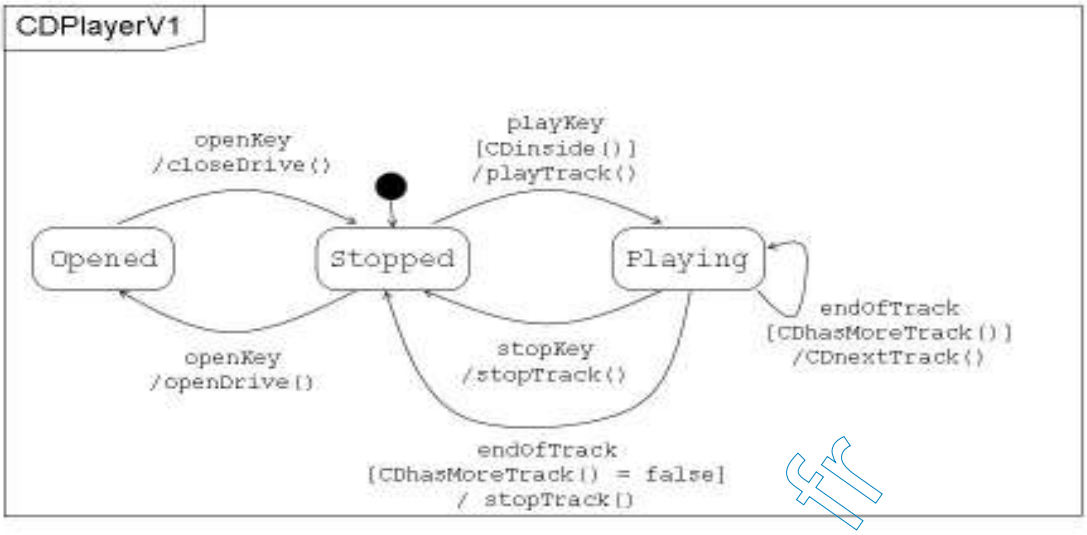
http://www.netbeans.org

Une fois l'installation terminée, allez dans le menu « tools / plugin » de NetBeans, pour installer le plugin « UML » (ATTENTION : dans cette fenêtre, il faut peut-être configurer votre proxy).

1 : Objectifs du TD/TP

Grâce aux outils de MDA fournis par NetBeans, nous allons développer un lecteur CD simple (*CD Player*), composé de trois boutons :

Le comportement associé à la version 1 de cette application correspond au diagramme d'état UML suivant :



Rappel : Une transition est définie par un état d'entrée, un message, une éventuelle condition, une action et un état de sortie.
Exemple : Dans l'état d'entrée Playing pour le message endOfTrack et si l'appel CDhasMoreTrack() renvoie false alors on exécute l'action stopTrack() et on passe dans l'état Stopped.

L'interface du *CD Player* doit permettre de traiter les événements provenant des boutons (`openkey`, `playkey` et `stopkey`) et l'événement `endOfTrack` qui signifie la fin d'une piste.

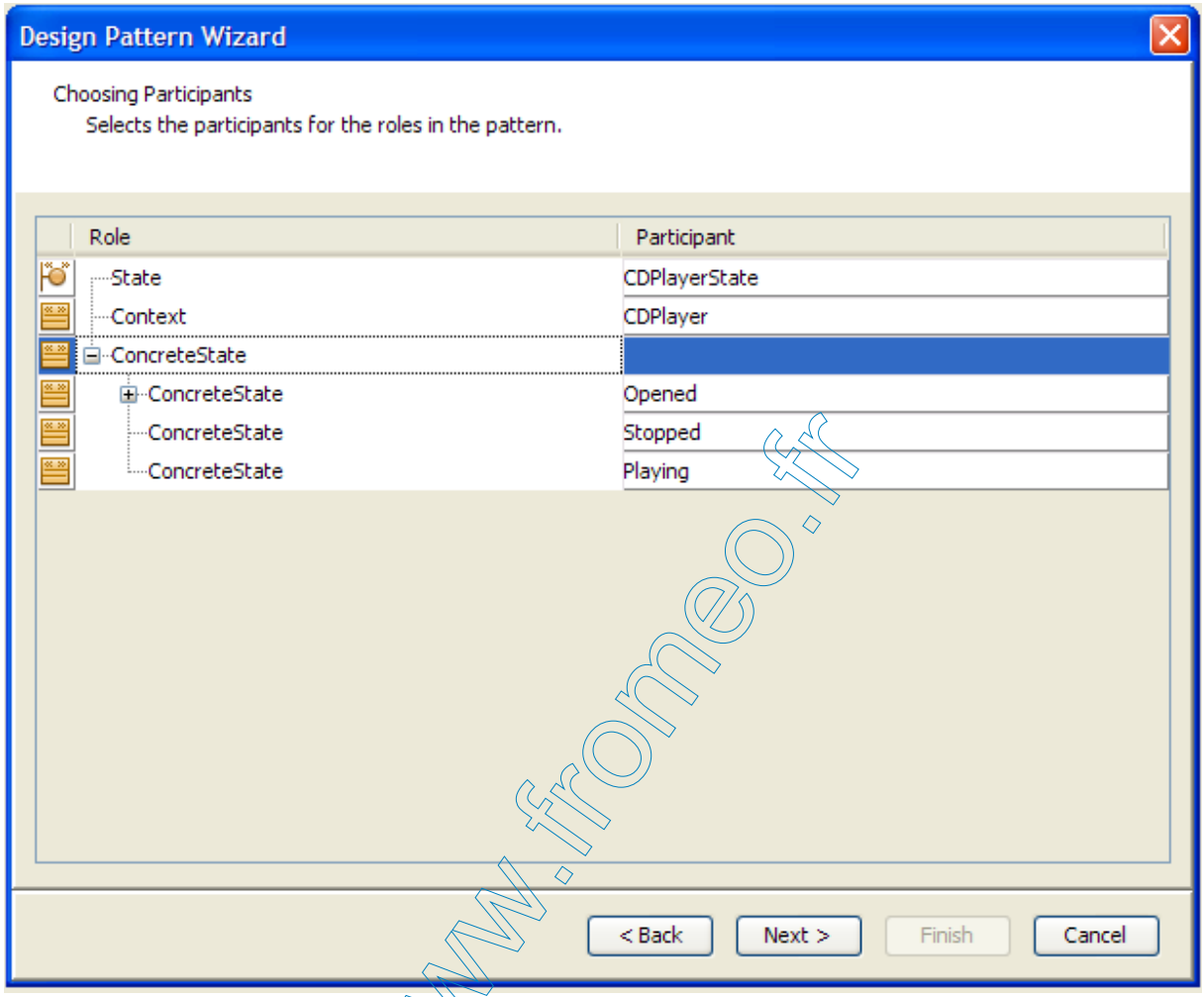
Le développement de cette application se fera selon le cycle de développement suivant :

- Modélisation assistée avec les patterns du GoF (définition d'un PSM)
- Génération automatique de code Java

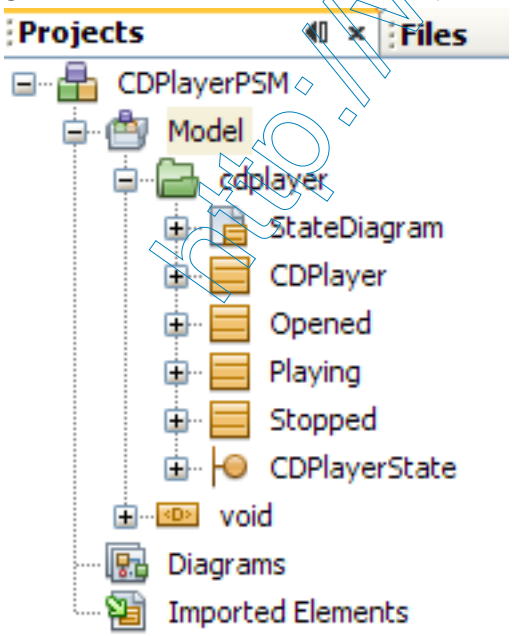
2 : Modélisation assistée avec les patterns du GoF

1. Créez un nouveau projet de type « **UML / Java Platform Model** » et nommez le « **CDPlayerPSM** »
2. NetBeans vous demande alors de créer un nouveau diagramme, laissez par défaut et appuyez sur « **Finish** »
3. Dans le panneau « **Projects** », vous pouvez voir le contenu de votre projet
4. Dans votre projet, faîtes un clic droit **sur le « model »** pour afficher les options : créez un package « **cdplayer** »
5. Choisissez maintenant sur le « **model** » l'option « **Apply Design Pattern** » : le Wizard se lance. Complétez alors les renseignements suivants :
    - Project : **Gof Design Patterns**
    - Design Pattern : **Behavioral ::State**
    - (Next)
    - Namespace : **cdplayer** (c'est le package précédemment créé)

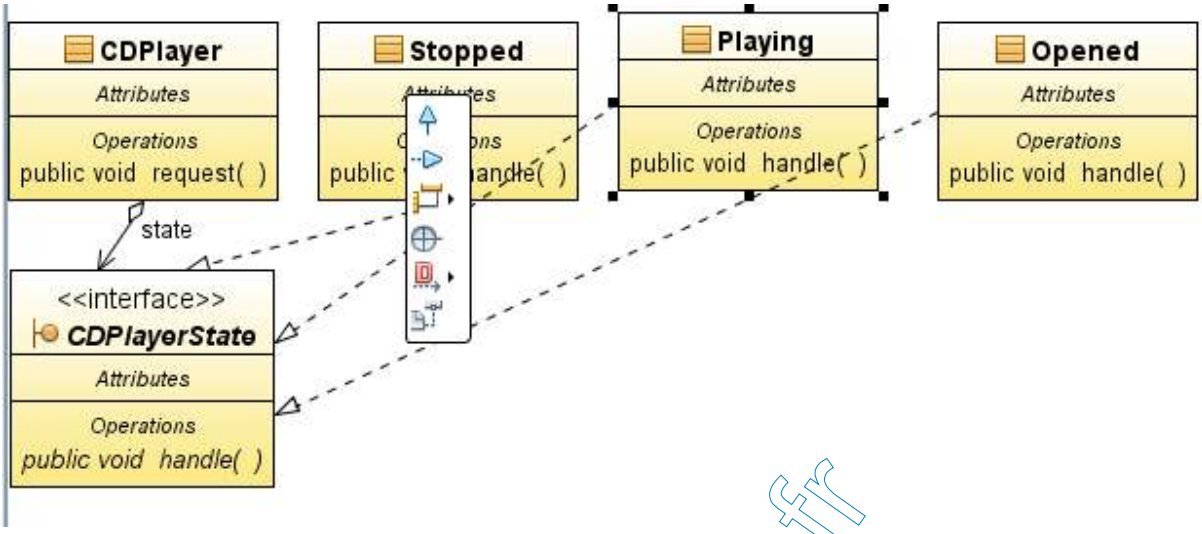
6. Complétez ensuite la structure du design pattern comme suit :



7. Cochez la case « **Create class diagram** » puis cliquez sur Next et Finish. Le modèle suivant est généré :

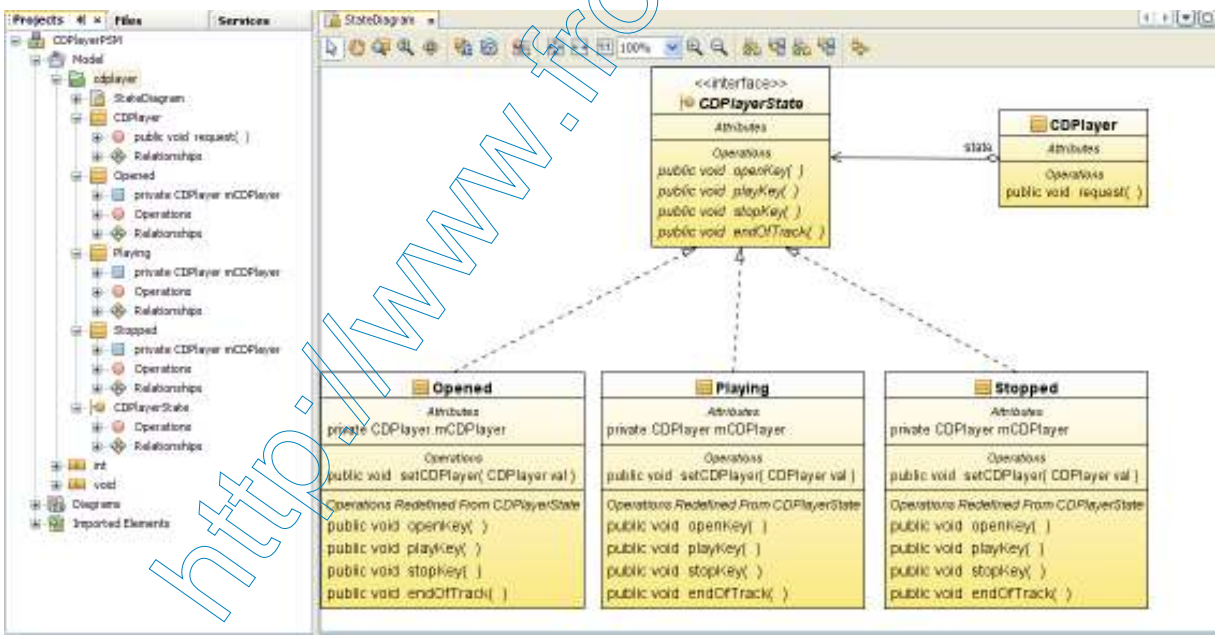


ainsi que le diagramme de classe correspondant :



Vous pouvez mettre en forme le diagramme de façon automatique avec le bouton « Hierarchical Layout » (Ctrl+Maj+K). C'est un outil pratique mais dans bien des cas il est préférable de le faire manuellement. Les algorithmes de placement des diagrammes sont encore un problème ouvert.

8. Modifier le modèle de votre application pour obtenir :

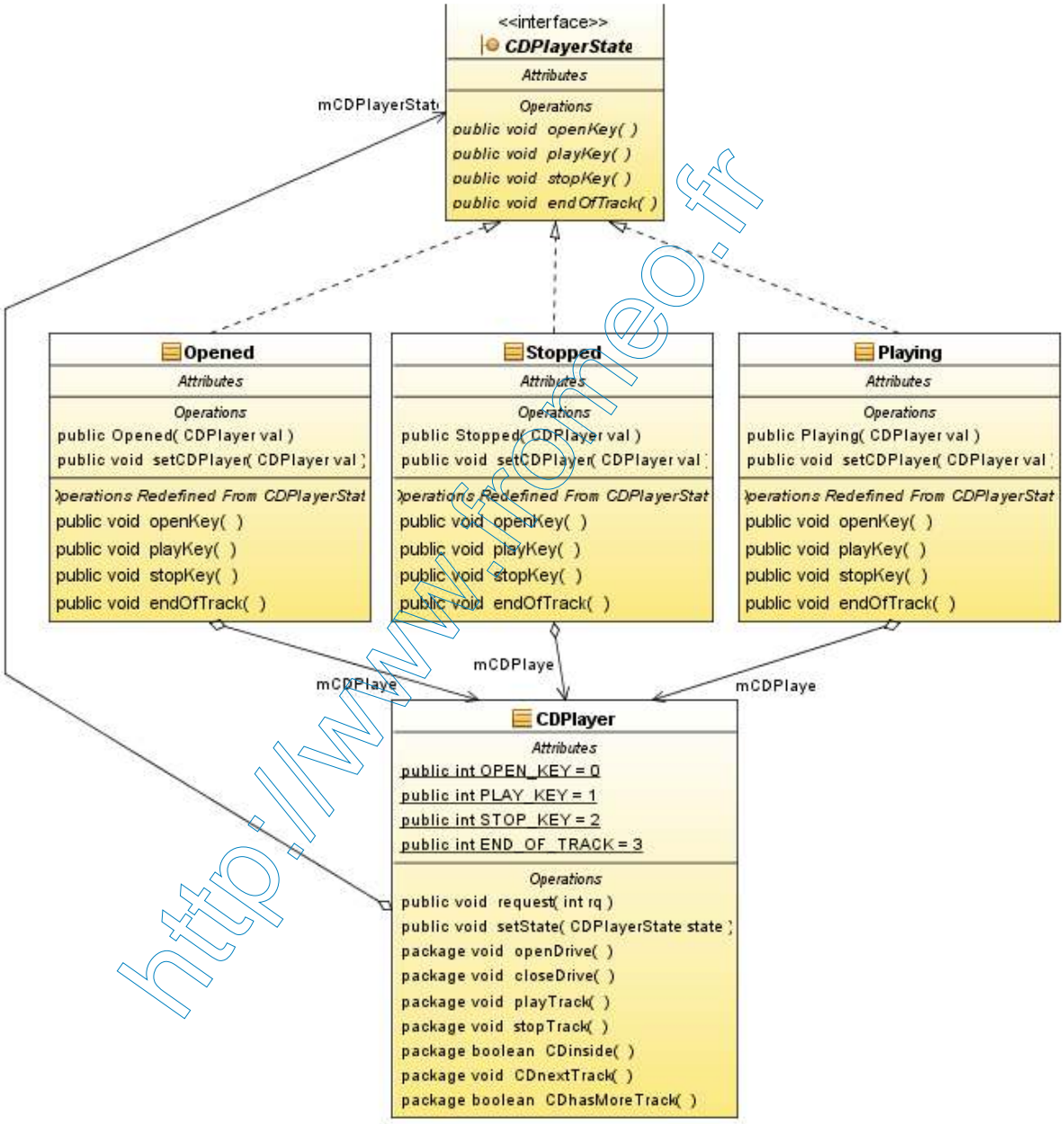


3 : Génération automatique de code Java

9. Créez maintenant un nouveau projet « **Java / Java Application** » : nommez le « **CDPlayer** »
10. Générez ensuite le code avec un clic droit sur le **projet UML CDPlayerPSM** choisissez l'option « **Generate Code…** ». Remplissez le projet cible Java (Target Project : **CDPlayer)**, puis OK. Le code est généré.
11. Complétez le code de la classe **CDPlayer** dans le **projet Java** avec la méthode void request(int rq) pour l'envoi des requêtes sur le CDPlayer. Vous utiliserez une instruction *switch/case* pour l'implémenter, avec un *case* par événement que vous créerez comme champs statiques de la classe :
    - final static public int OPEN_KEY = 0 ;
    - final static public int PLAY_KEY = 1 ;
    - final static public int STOP_KEY = 2 ;
    - final static public int END_OF_TRACK = 3 ;
12. Implémentez les autres méthodes de la classe CDPlayer, c'est-à-dire les méthodes setState, openDrive, closeDrive, playTrack, stopTrack, CDinside, CDnextTrack, CDhasMoreTrack.
    Par exemple :
    - Pour closeDrive : System.out.println("closeDrive")
    - Pour CDinside : return true
    - Pour CDhasMoreTrack : return false
13. Codez ensuite les classes des états concrets avec les transitions correspondant à la machine à états UML CDPlayerV1

4 : Rétro-ingénierie

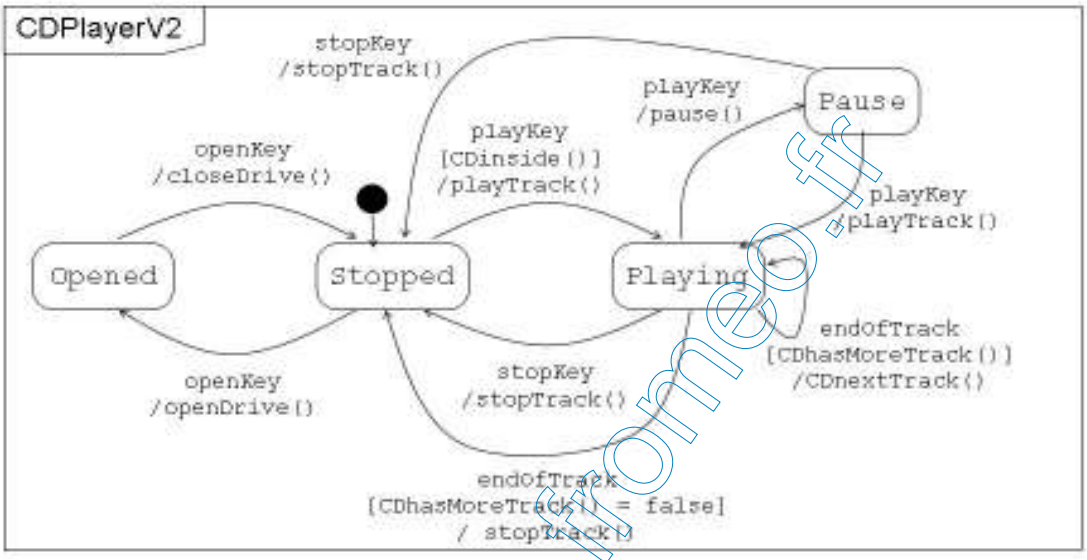
14. Sur projet Java CDPlayer, faîtes un clic droit pour lancer la rétro-ingénierie « Reverse Engineering ». Choisissez le même projet UML, CDPlayerPSM puis Yes to All.



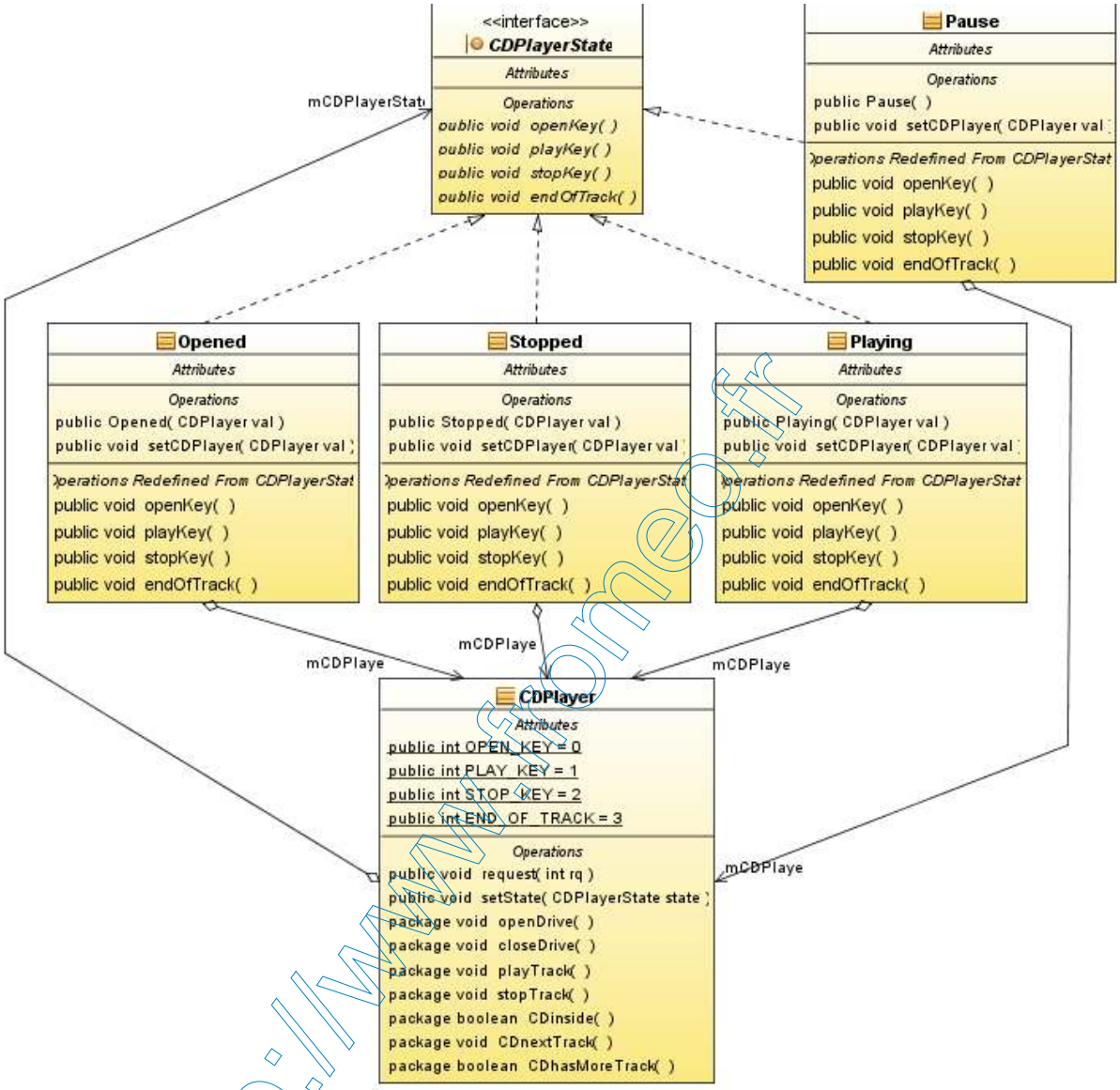
Après remise en forme, vous obtenez le diagramme de classe ci-dessus.

5 : Nouvelle itération sur le cycle de développement

Sur la base de la version V1, on veut maintenant réaliser une nouvelle version V2 de cette application qui intègre la possibilité de mettre le lecteur sur « Pause » grâce au bouton de lecture (`playkey`). Voici la spécification de ce nouveau comportement en UML :



15. Réalisez les modifications suivantes sur votre modèle dans le projet CDPlayerPSM :



16. Générez à nouveau le code puis complétez votre application.