Grapheur

Document d'architecture et de conception

Réalisé par : GEHIER Kylian

Encadré par : Jean-Marc Perronne – Enseignant chercheur à l'ENSISA

Table des matières

1 Présentation générale du projet

Le projet présenté dans ce document d'architecture et de conception est une API ayant pour but l'affichage de Graphes (Charts en anglais).

Ce projet m'a été attribué par Jean-Marc Perronne comme projet de fin d'étude pour mon diplôme d'ingénieur informatique et réseau.

1.1 Une première conception du projet (2012)

Le projet Grapheur à vu le jour en 2012. Sa conception avait été confiée à Jean-Marc Perronne par l'entreprise INFRAL pour de la télémédecine.

Le projet avait finalement été interrompu pour des raisons qui me sont inconnues.

Avant son interruption, le Grapheur était alors capable :

- D'afficher deux types de graphes qui seront détaillés plus tard dans le document : les LineChart et les MapChart
- De lire un fichier Audio statiquement ou de simuler une acquisition en temps réel depuis ce même fichier audio
- D'enregistrer des données depuis un Micro puis de les ajouter statiquement ou dynamiquement à un Chart.

1.2 Reprise du projet (2018)

Dans le cadre d'une projet de troisième année d'école d'ingénieur, il m'a été demandé par Jean-Marc Perronne de reprendre ce Grapheur et d'y apporter diverses modifications listées dans le cahier des charges ci-dessous.

Le projet ayant été avorté en 2012, ce dernier n'avait pas été documenté. Il m'a donc fallut, dans un premier temps, me familiariser avec l'API et en comprendre les rouages.

1.3 Cahier des charges

Cahier des charges imposé :

- Dés-Infraliser le projet : nettoyer toute trace d'Infral présente dans le projet
- Généraliser le Grapheur :
  - permettre l'ajout de donnée statique venant d'un CSV
  - implémenter une Factory (Usine à Graphe) pour simplifier l'utilisation du Grapheur
- Nettoyer le code (Prints, commentaires) et les packages du projet
- Rédaction d'un document de conception et d'architecture (Ce document)

Mes propositions validées par M.Perronne :

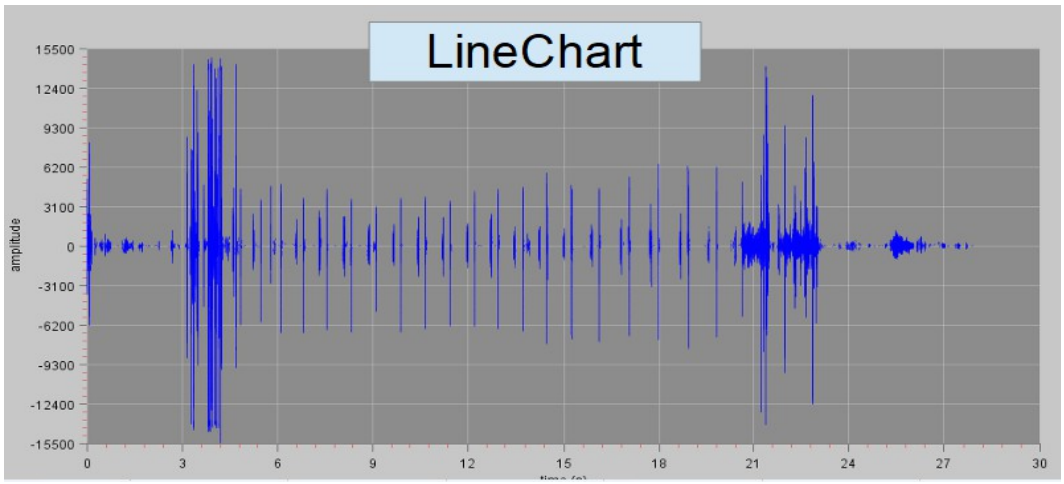
- Implémenter un système de fichier de configuration pour le paramétrage des Charts.
- Création d'un modèle d'abstraction « Entry » regroupant les différents types de données entrantes afin de faciliter l'ajout de données sur un Chart.

2 Architecture

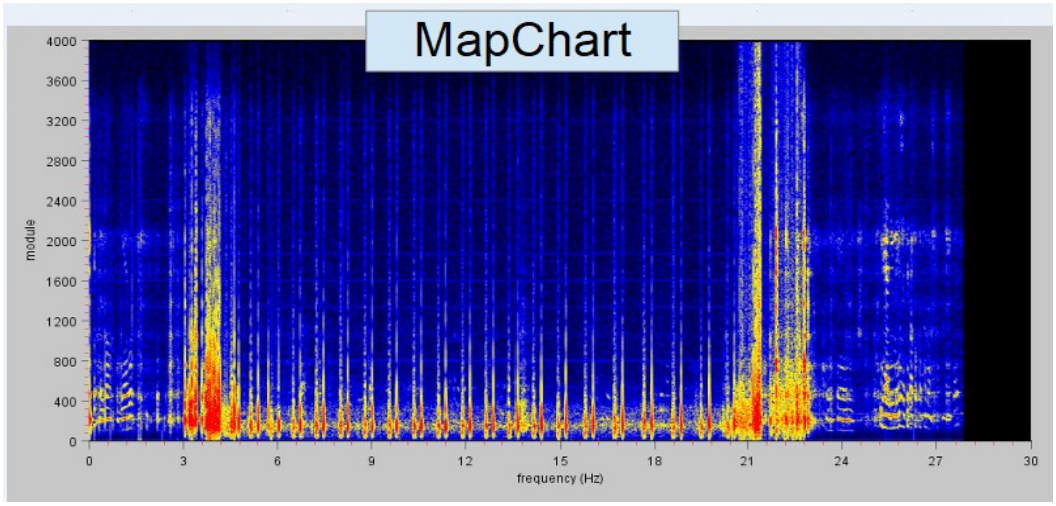
L'une des particularités de ce Grapheur est qu'il se base sur un système d'axes. Cette particularité permet la création de différents types de Chart au sein du même projet.

On distingue à l'heure actuelle deux types de Chart que sont les LineCharts et les MapCharts.

- Un LineChart est un Graphe 2D dans un repère orthonormé :



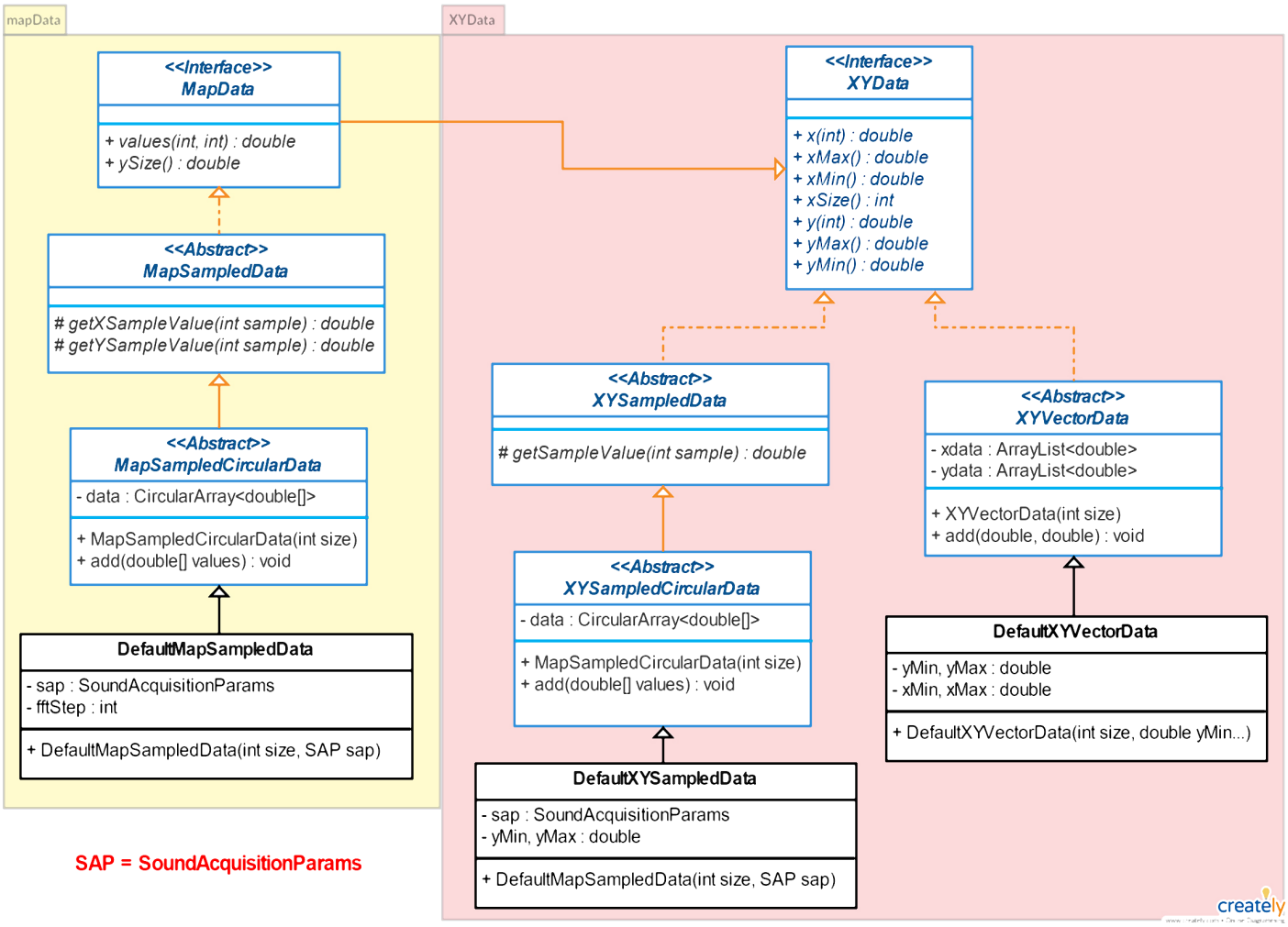
- Un MapChart est une Graphe 2D comprenant un champ de profondeur colorimétrique donnant un aspect de Graphe 3D



2.1 Modèle de données

Les Charts utilisent un modèle de données qui leur est propre :

Diagramme de classe – Modèle de données



Comme le montre le diagramme de classe ce-dessus, il existe deux interfaces principales :

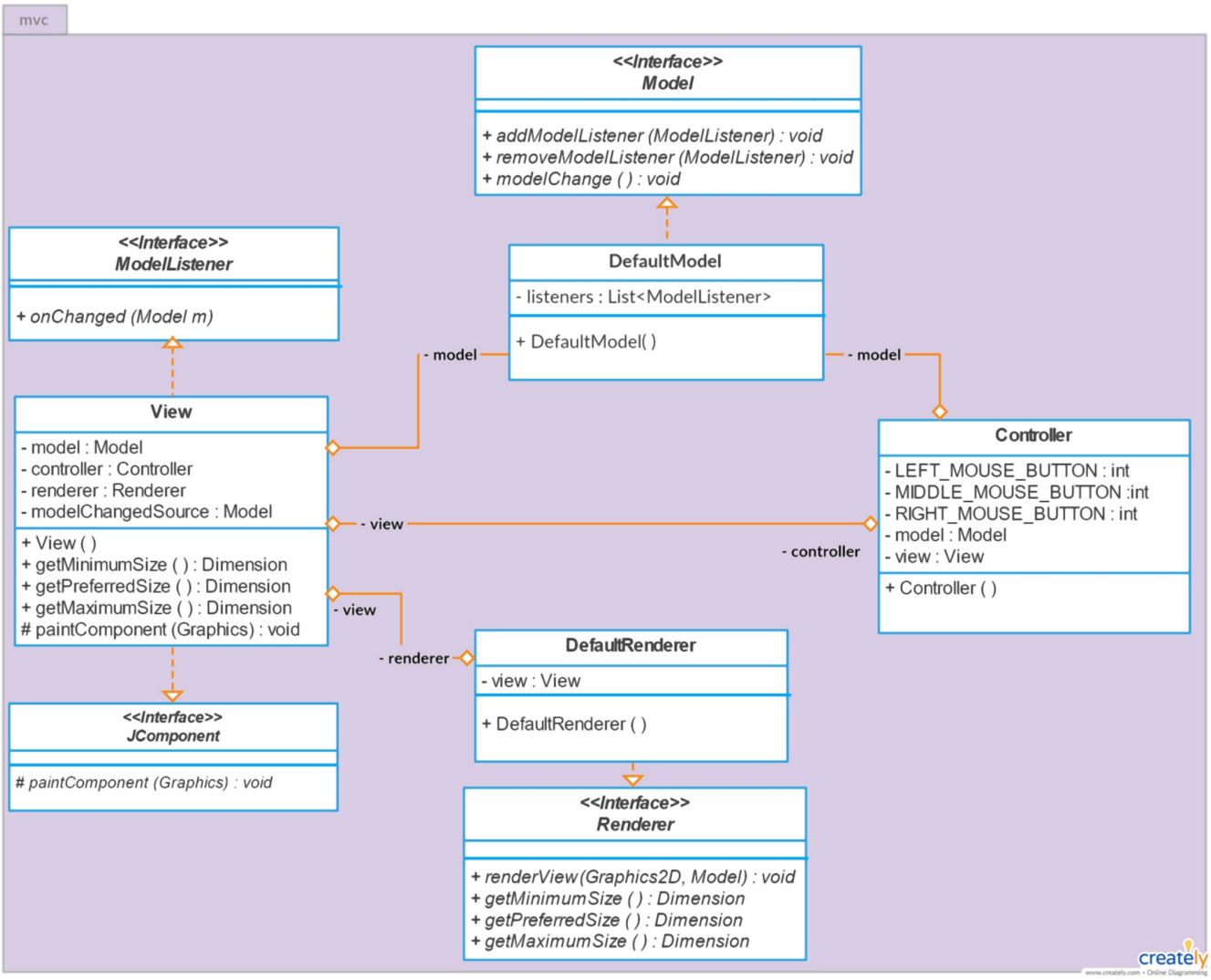
- XYData : modèle de données utilisé par les LineChart
  - XYSampledData : utilisé pour des données échantillonnées (car cela implique un intervalle constant dx)
  - XYVectorData : utilisé pour des données non-échantillonnées (ce qui implique le besoin d'utiliser un tableau de donnée en x)

- MapData : modèle de données utilisé par les MapChart
  - MapSampledData : utilisé pour des données échantillonnées
  - MapVectorData : il n'y pas de cas d'utilisation impliquant des données non-échantillonnées dans ce document. La classe n'a donc pas encore été implémentée.

2.2 MVC

Notre Grapheur repose sur une architecture MVC (Modèle – Vue – Contrôleur) Swing à laquelle vient s'ajouter un Renderer.

Diagramme de classe – MVC

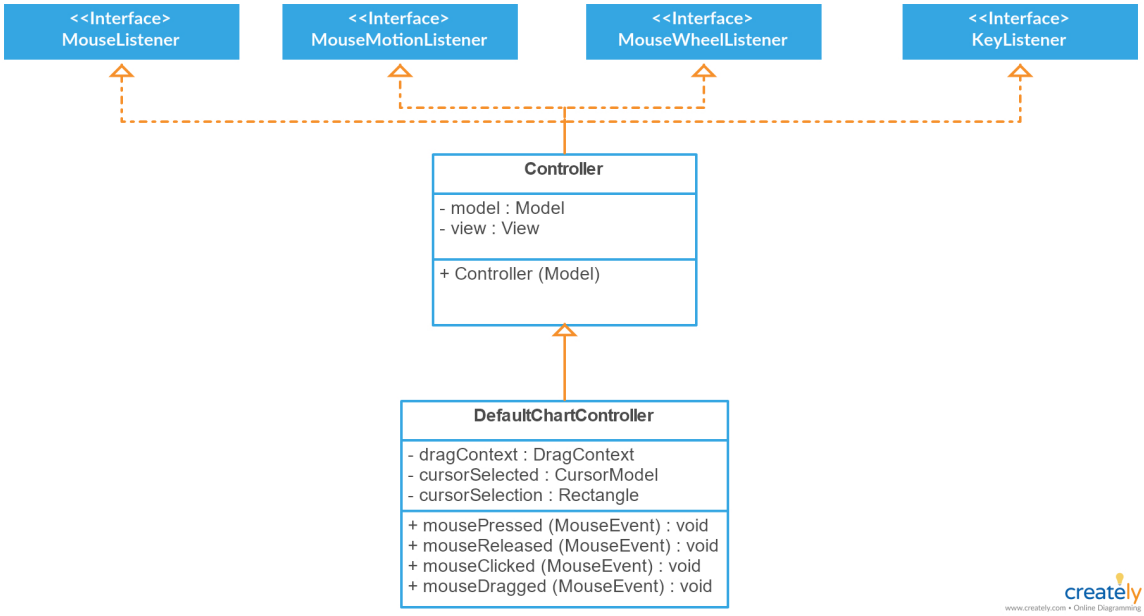


2.1 Description des différents composants du MVC

- **Model (Modèle)** : contient les données du Chart sous une forme compréhensible par la Vue et la notifie lorsque des changements lui sont appliqués.

- **Controller (contrôleur)** : réceptionne les différentes actions effectuées depuis la Vue grâce aux Listener qu'il Implémente (cf diagramme ci-dessous) puis modifie le modèle en conséquence.

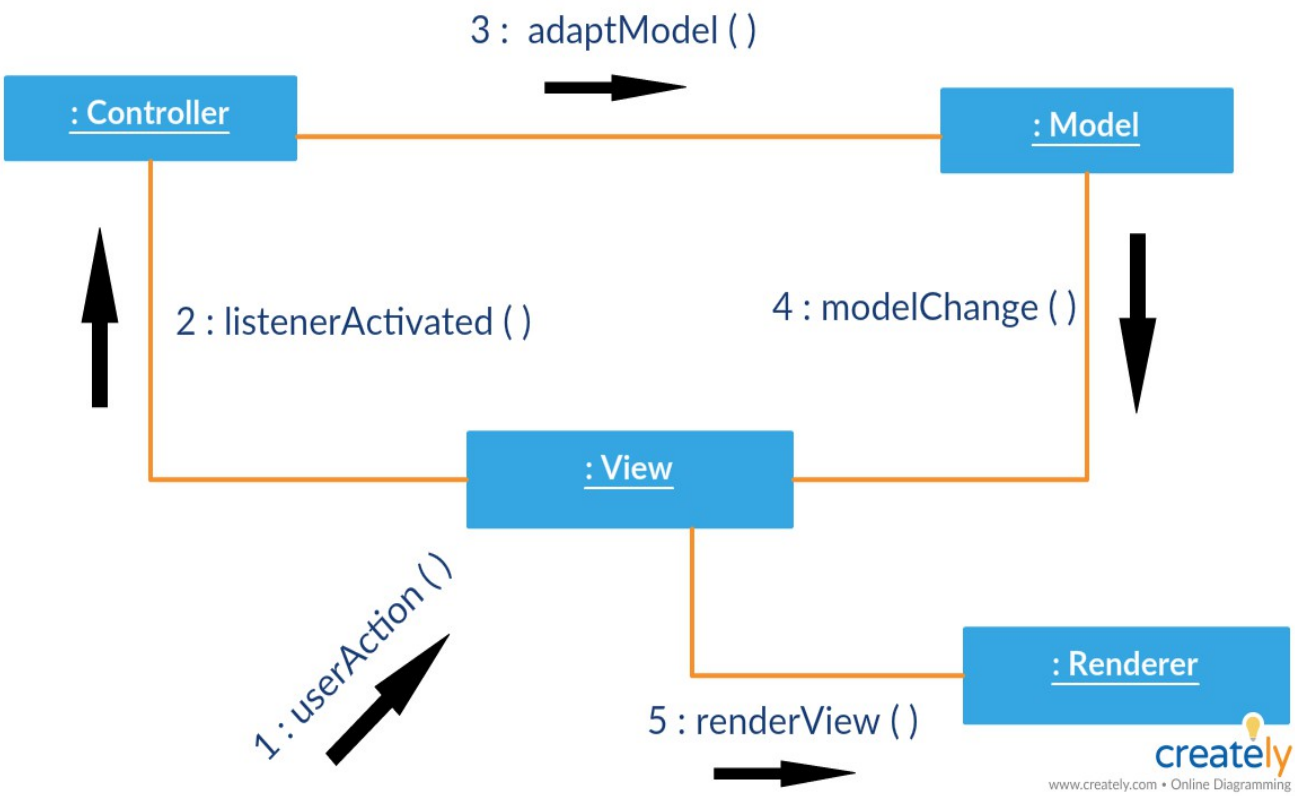
Diagramme de classe – Controller



- **Renderer** : chargé par la Vue de dessiner les modèles en récupérant leur données et leurs attributs.

- **View (Vue)** : c'est l'IHM (Interface Humain Machine). Les modèles y sont dessinés par les renderers associés à chaque modèle. La Vue est donc le Chart.

Diagramme de communication – Action d'un utilisateur sur la Vue



Comme le montre le diagramme ci-dessus, lorsqu'un utilisateur effectue une action depuis la Vue (ex : click de souris), cette dernière active un ou plusieurs Listener(s) (écouteur) implémenté(s) par le contrôleur.

Le contrôleur modifie alors le ou les modèle(s) concerné(s) qui va/vont a leur tour notifier la Vue qu'il(s) a/ont été modifié(s) via la méthode « modelChange » commune à tous les modèles.

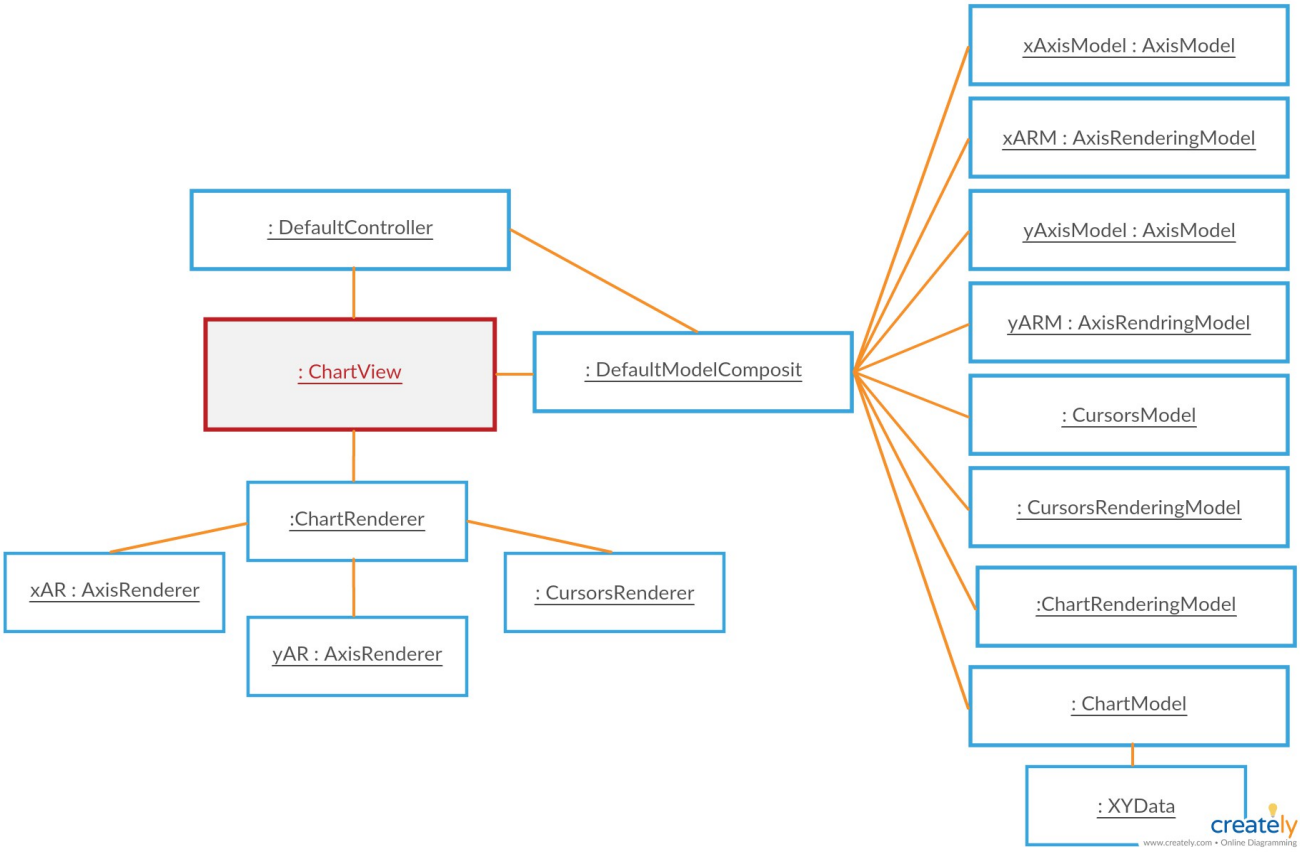
La Vue appel alors sa méthode « paintComponent » (Cf. Diagramme de classe MVC) qui délégue le dessin de tous les modèles au renderers.

2.2 Les composants d'un Chart

Nous venons de voir qu'un Chart était en fait une Vue décliné du MVC.

Je parlerais donc maintenant de « ChartView » pour désigner un Chart.

Diagramme d'objets– Composition d'un Chart



Comme nous pouvons le voir sur le diagramme ce-dessus, un ChartView est donc composé de deux modèles d'axes (x et y), d'une modèle de curseurs et de son modèle de Graphe. Ces différents modèles seront détaillés plus loin.

Il est également composé d'un modèle de rendu (RenderingModel) pour chaque modèle contenant les données nécessaire à l'affichage graphique de chacun.

Chaque modèle se voit également attribué un Renderer qui sera chargé de le dessiner en récupérant les informations contenu dans les Model et RenderingModel associés.

Un model peut donc être considéré comme un triplet : (Model, RenderingModel, Renderer).

Nous pouvons également remarqué la présence d'un pattern Composite qui sera également détaillé plus loin dans le document.

Description des différents composants du MVC :

- Model (Modèle)
  - Contient les différents modèles composant le Chart
  - Mis sous la forme d'un « pattern Composite » pour faciliter l'accès aux différents Modèles depuis les différents niveaux d'abstraction de la Vue

3 Cas d'utilisations