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 :

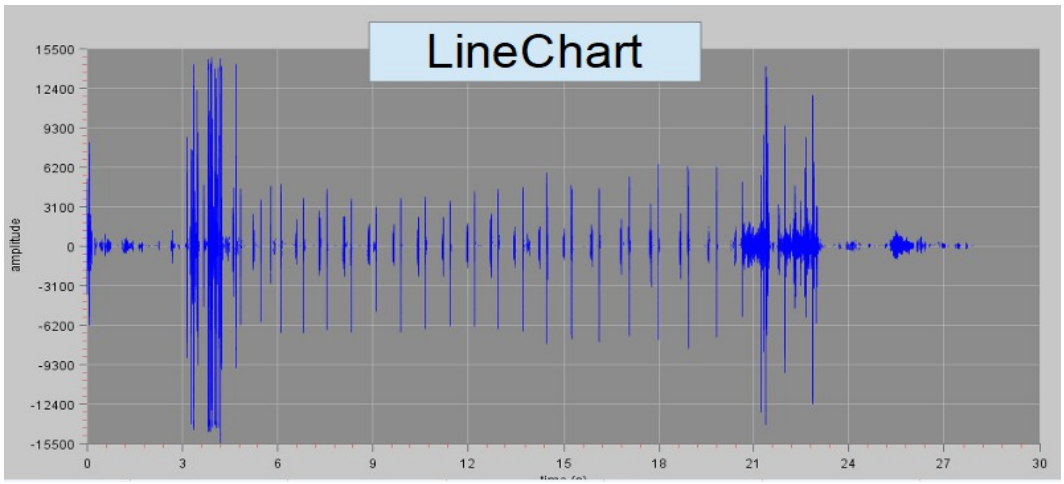
- Implémenter un système de fichier de configuration pour le paramétrage des Charts.
- Création d'un modèle d'abstraction « Entry » regroupant les différents types de données entrantes afin de faciliter l'ajout de données sur un Chart.

2 Architecture

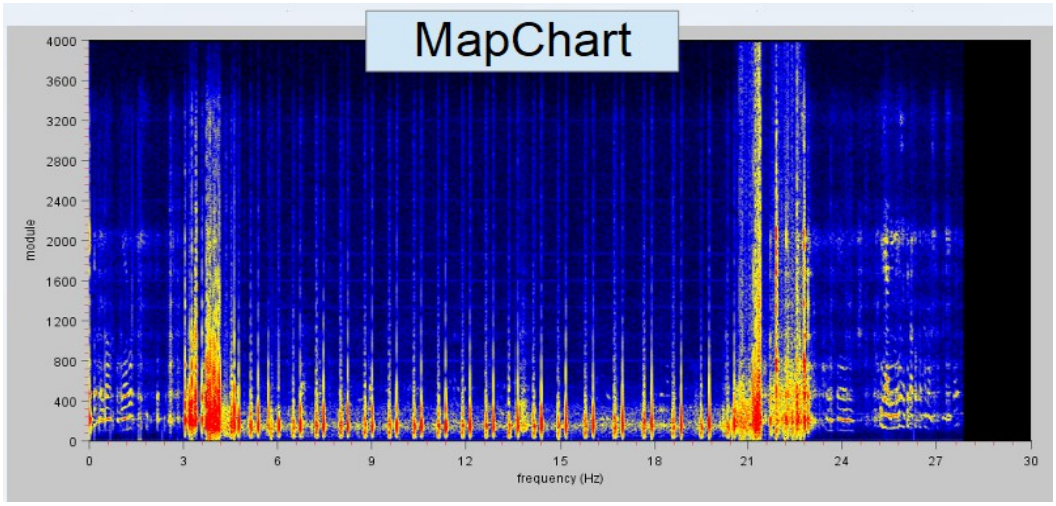
L'une des particularités de ce Grapheur est qu'il se base sur un système d'axes. Cette particularité permet la création de différents types de Chart dans ce même projet.

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 :



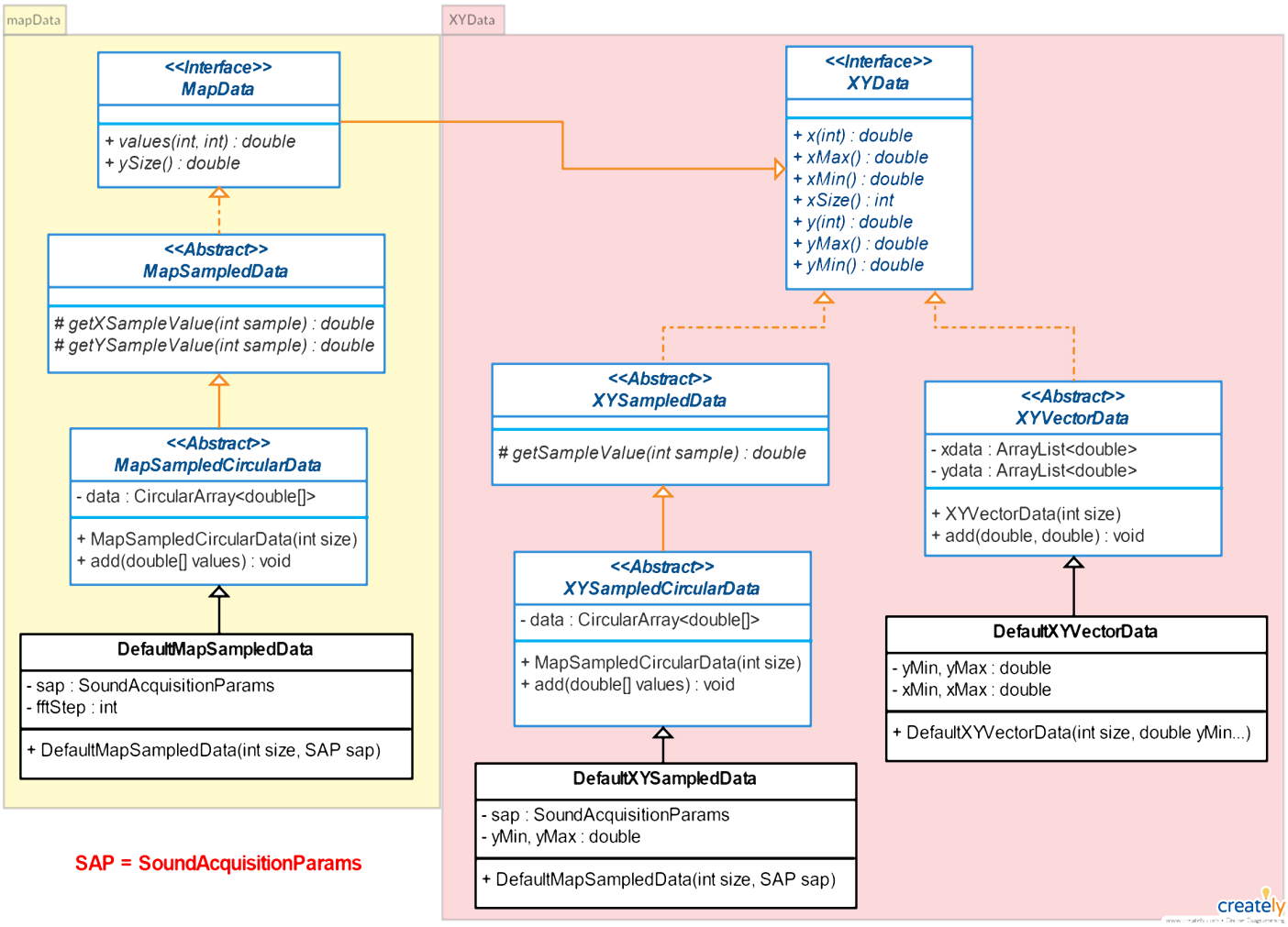
- Un MapChart est une Graphe 2D comprenant un champ de profondeur colorimétrique donnant un aspect de Graphe 3D



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 :

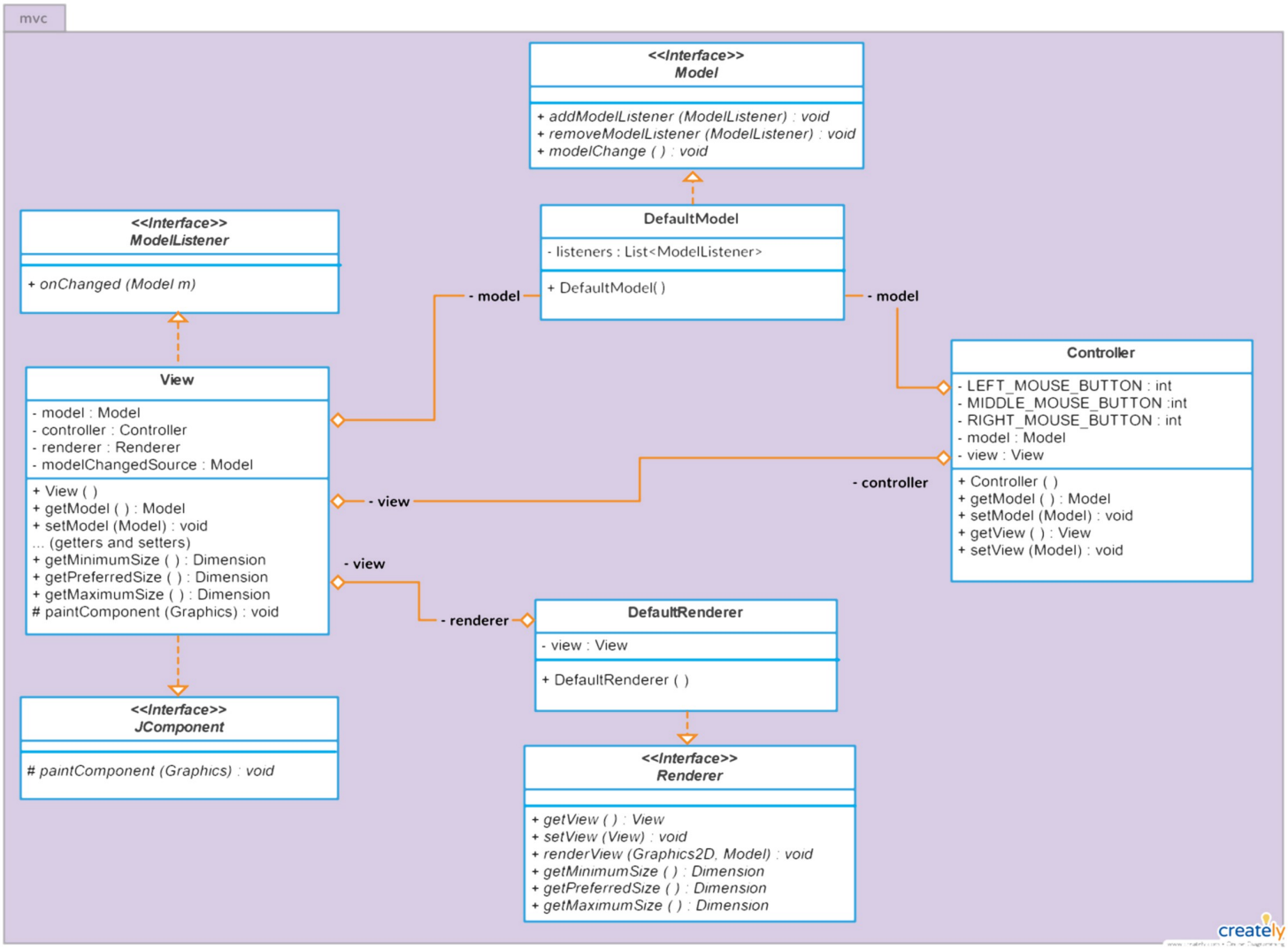
- XYData : modèle de données utilisé par les LineChart
  - XYSampledData : utilisé pour des données échantillonnées (car cela implique un intervalle constant dx)
  - XYVectorData : utilisé pour des données non-échantillonnées (ce qui implique le besoin d'utiliser un tableau de donnée en x)

- 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 été créée.

2.2 MVC

L'architecture sur laquelle repose le Grapheur est une architecture MVC – Swing à laquelle a été ajouté un module chargé du rendu des différentes composantes d'un Chart : le Renderer.

Diagramme de classe - MVC



Comme pour une architecture MVC-Swing classique :

- Le Modèle contient les données du Chart sous une forme compréhensible par la Vue et la notifie lorsque des changements lui sont apportés.
- Le contrôleur réceptionne les actions effectuées depuis la Vue et modifie le Modèle en conséquence.
- La Vue est l'interface graphique affichant le Modèle.

L'ajout du Renderer permet à la Vue de lui déléguer la tâche du dessin des Modèles.

2.3 Modèle

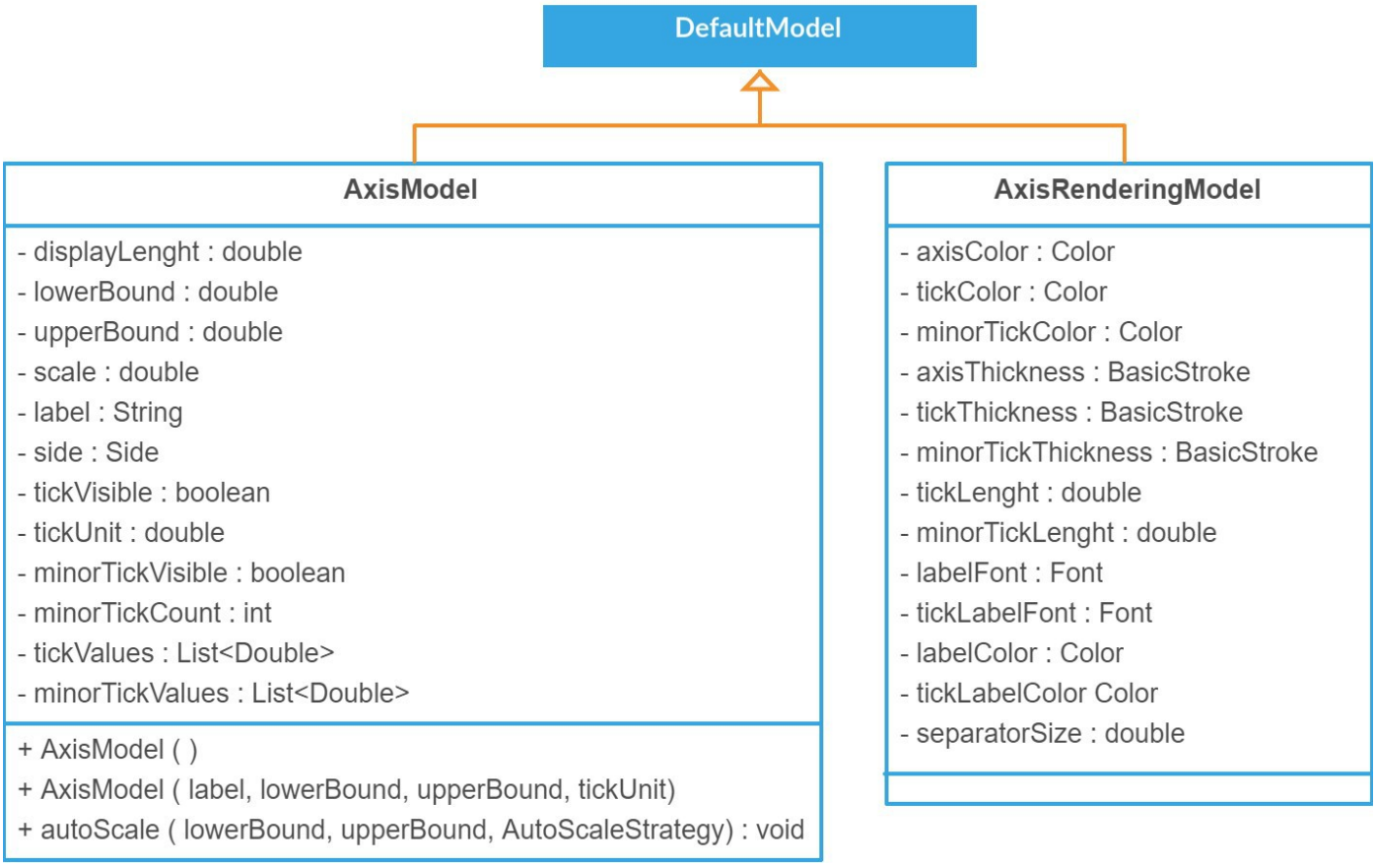
Il est important de noter que chaque modèle est scindé en un couple de Model/RenderingModel (ou Modèle/Modèle de Rendu) :

- Model : contient les informations de visibilité des composants du modèle, les données, …
- RenderingModel : contient toutes les informations relatives à l'affichage graphique du modèle

On distingue trois « familles » de modèle :

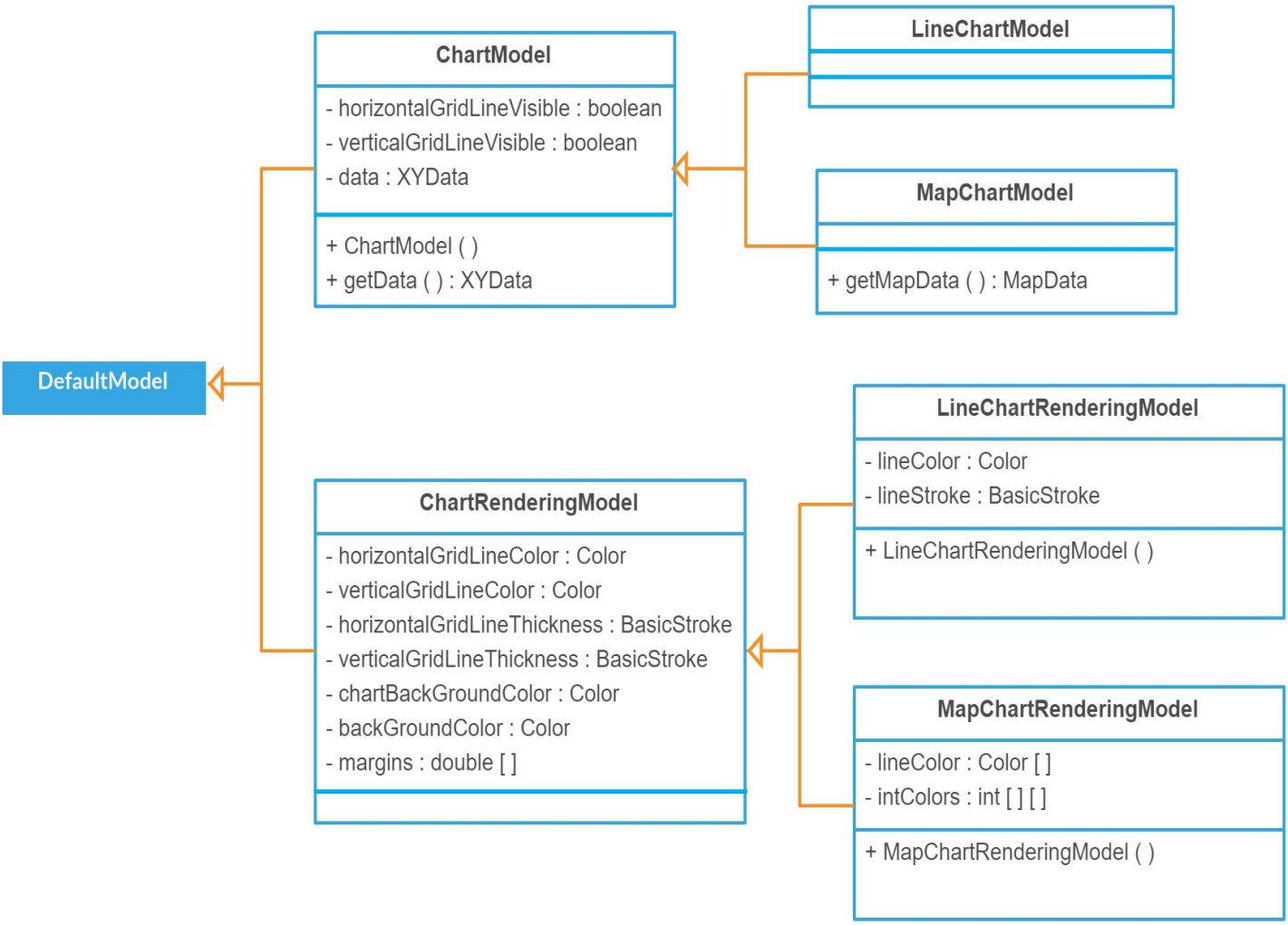
- AxisModel / AxisRenderingModel :
  - modèle autour duquel le Chart se construit comme expliqué précédemment
  - contiennent les informations relatives à l'affichage des axes du Chart

Diagramme de classe – AxisModel / AxisRenderingModel



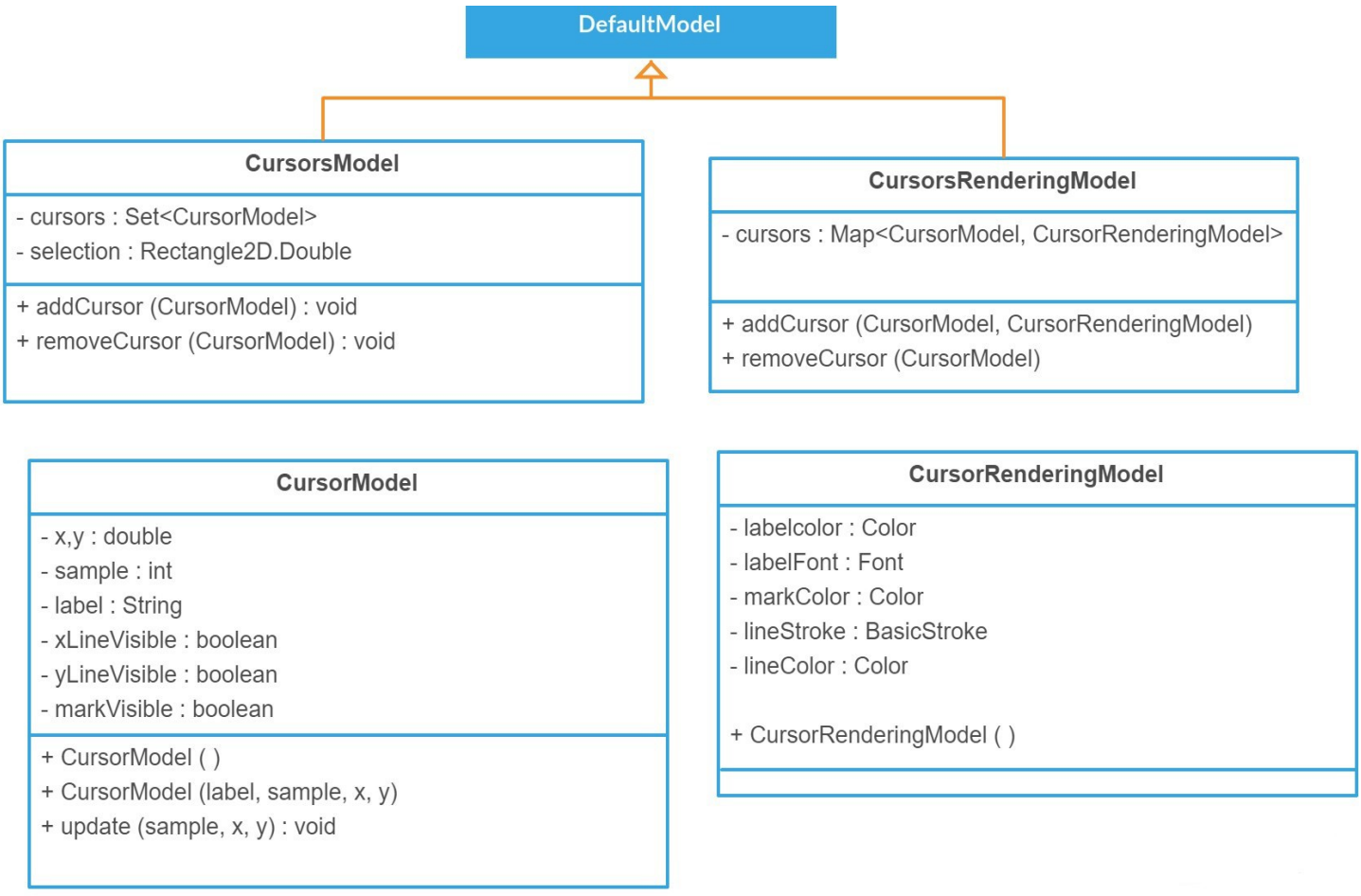
- ChartModel (Modèle de Graphe)
  - modèle contenant les informations relatives à l'affichage du Chart
  - C'est dans le ChartModel que sont stockées les données du Chart (XYData / MapData)

Diagramme de classe – ChartModel / ChartRenderingModel



- CursorsModel (Modèle de Curseurs)
  - modèle contenant les informations relatives à l'affichage des curseurs
  - ces curseurs peuvent être insérés par l'utilisateur depuis la Vue (cf 2.5 Controlleur)

Diagramme de classe – CursorsModel / CursorsRenderingModel



3 Cas d'utilisations