# Mise en place des classes de données

## Présentation

### Document concerné

Ce document traite des classes de données contenues dans un neurone.

Je rédige ce court document pour préciser le travail de programmation nécessaire.

### Versions

06/12/2016 : version initiale

### Auteur

Invisible Media

## Projet traité

### Cadre du projet

Le présent projet est un logiciel de calcul numérique et algébrique.  
Ce projet est un moyen de former des équations très longues et fastidieuses et, de les factoriser selon une approche commune.

But : résoudre des équations, trouver l’ensemble des solutions algébriquement et obtenir une équation en fonction de paramètres et de données numériques fixées.

### Ensemble des fonctionnalités

L’ensemble des fonctionnalités est détaillé dans ce document :

[Toutes les fonctionnalités (dégroupé).xlsx](Toutes%20les%20fonctionnalités%20(dégroupé).xlsx)

### Nom de la fonctionnalité traitée ici

Classes des données fournies à chaque neurone

## Besoin relatif

Le codage des noms a une importance de lisibilité. Les noms sont référencés par des variables de classes constantes nommées et qui contiennent un nom.   
Associé à ce nom, j’ajoute l’information du type de données requis.   
Si la donnée nommée ne contient pas le bon type de données alors une erreur est déclenchée.

Pour toute donnée enregistrée dans un objet persistant (via une sérialisation binaire), le couple nom/type contient une instance de la donnée à sérialiser.

Retrouver cette donnée est possible uniquement par le nom.

Il est indispensable de consigner dans ce document tout nom associé à chaque donnée prévue dans le modèle de données.

## Relation avec les autres fonctionnalités

### Utilisation

La donnée est consignée, lue et modifiée au cours du temps via le modèle de données

### Explications

Il s’agit de savoir traiter toutes les versions du modèle de données.

Or, une sérialisation binaire implique que toutes les données sont inscrites dans le fichier persistant.

Une fois ce fichier sauvegardé sur disque, pour le reprendre, il suffit de dé-sérialiser.

Excepté que si le logiciel est à une version ultérieure, toutes les données inexistantes dans le fichier déclenche une erreur de donnée **null** qu’il est difficile de tenir compte. Dans ce cas, le logiciel devrait revenir à une version précédente.

Lorsque le système utilise la version antérieure, il encadre également l’ensemble des versions ultérieures.

A noter que chaque fichier comporte un numéro de version du logiciel ; si un fichier n’est pas de la bonne version, ce problème est corrigé une fois le fichier écrasé et sauvegardé avec la nouvelle version.

## Programmation

Je pense qu’il est plus pertinent d’avoir une classe de données intermédiaires entre la dé-sérialisation et les objets métiers.

Le processus de lecture est décrit dans le schéma ci-dessous expliqué ci-après.

Lors de la dé-sérialisation, un dictionnaire de données est reconstruit avec le type *dynamic* du dotNET. Le type est pré-établi depuis une classe statique nommée pour l’objet concerné. Comme le type de donnée est de type *dynamic*, il est inconnu au moment de la compilation ; une fois l’objet chargé, il suffit de construire une classe intermédiaire dont les propriétés sont en lecture/écriture et dont le nom de chaque propriété est spécifique. La classe intermédiaire héberge donc l’objet dé-sérialisé et peut récupérer ou modifier la valeur contenu dans le dictionnaire de donnée.

Lorsque je sauvegarde les modifications par sérialisation, le dictionnaire de données est enregistré dans le fichier. Les classes intermédiaires et les instances de neurone traitent les données contenues par ces classes. Ces classes intermédiaires héberge les objets sérialisés et aucune autre données.

Aussi, pour la clarté du code, les classes intermédiaires implémentent également des types simples tels que du texte ou des nombres. Dans ce cas, il s’agira de conserver l’objet qui contient ces données simples et pas les données elles-mêmes.

En conclusion, toutes les classes intermédiaires sont publiques ; les objets sérialisés sont privés ; la sérialisation et dé-sérialisation s’effectuent en interne et, il n’existe qu’une seule classe de modélisation des objets. Il n’y a pas de traitement nécessaire selon que l’élément d’une donnée est un type simple ou un autre objet sérialisé. Dans le cas de listes ou de tableaux, il s’agit d’un type de donnée particulier capable d’enregistrer un ou plusieurs objets.