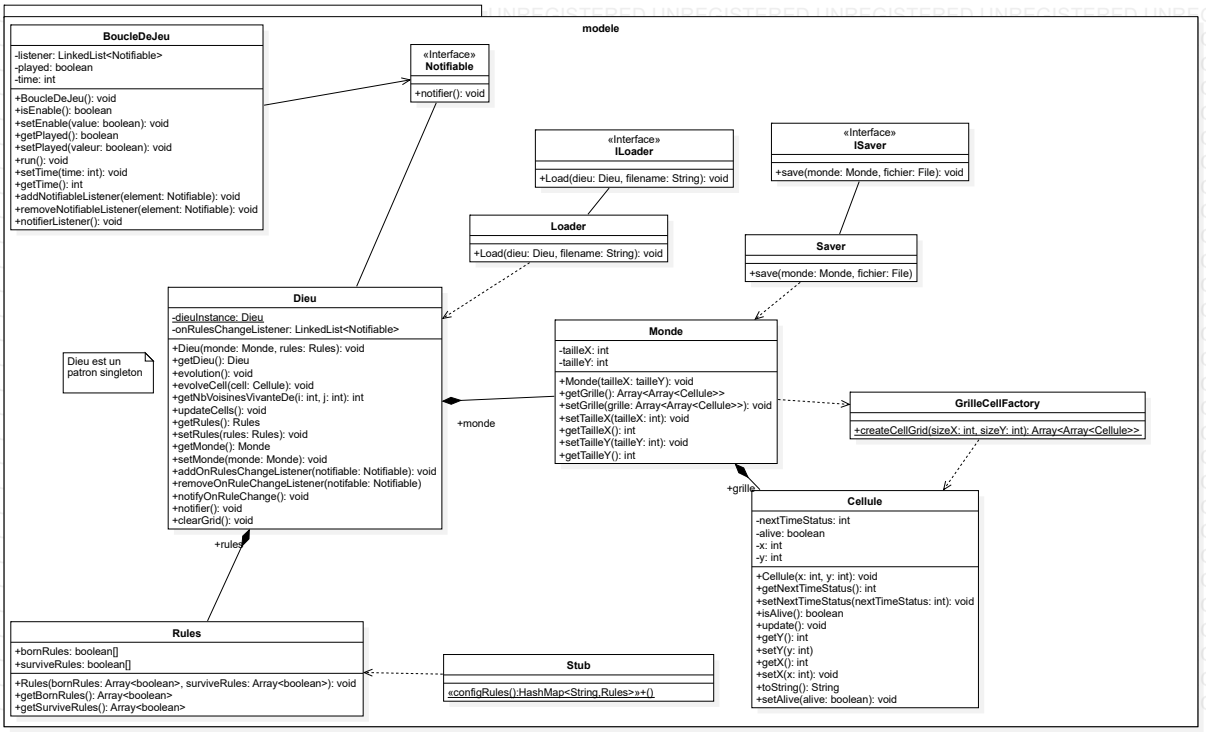
Explication diagramme de classe :



La classe *Rules* est la classe qui permet de créer nos règles pour nos cellules. Une règle est composée d'un tableau de règles de naissance et d'un tableau de règles de survie.

La classe cellule est la base de notre application. Une cellule est composée d'un entier nommé *nextTimeStatus*. Il correspond au statut que va avoir la cellule au prochain tour de boucle. Il peut prendre 3 valeurs différentes : -1 la cellule meurt, 0 la cellule ne change pas de statut, 1 la cellule naît. De plus, on a un booléen *alive qui représente le* statut actuel de la cellule, et les coordonnées *x* et *y* de la cellule dans la grille. La méthode *update()* change *alive en fonction de nextTimeStatus*.

La classe *GrilleCellFactory* est une classe qui créer un tableau de cellule à partir d'une taille. Cette taille est données par la *CustomView CellsGrid*, ce qui permet d'avoir une taille de grille adapté à l'écran.

La classe Monde est une classe qui prend en paramètre une taille de monde. C'est cette dernière qui va appeler la *GrilleCellFactory*.

La classe *Dieu* est le manager de notre application, et c'est un patron singleton. Il possède une *instanceDeDieu* (pour le singleton), une liste de *Rules*. C'est lui qui possède l'algorithme de l'application. La méthode *evolution()* parcours le tableau, et pour chaque cellule appel la méthode *evolveCell()*, qui détermine l'état d'une cellule en fonction des règles du monde, et du nombre des cellules vivantes adjacente à la cellule qu'on traite. Ce nombre est donné par la méthode *getNbVoisineVivanteDe()* à laquelle on donne les coordonnées de la cellule. La dernière méthode très importante de cette classe est la classe *updateCells()*. Cette dernière met toutes les cellules traitées à jour, c'est-à-dire qu'elle appel la méthode *update()* de chaque cellule du tableau pour qu'elle mettent à jour son statut. Cette classe hérite de l'interface *Notifiable*, pour que la boucle de jeu puisse notifier dieu et qu'il exécute l'algorithme dans sa méthode *notifier()*.

La classe *saver* est la classe qui permet d'enregistrer un motif dans un fichier. Pour enregistrer le motif, nous écrivons les coordonnées de chaque cellules vivantes de la grille sur une ligne du fichier, sous la forme « *x:y* ». La classe *saver* implémente l'interface *ISaver*. La méthode *saver()* prend en paramètre le monde et le chemin vers le fichier dans lequel écrire. Le système est le même pour le chargement, avec la classe *Loader* qui implémente *ILoader*.

La classe *stub* permet d'obtenir, via la méthode *configRules* une *HashMap* contenant des préréglages de règles, stocké sous forme d'instance de *Rules*. Il ne reste plus qu'à écraser les règles du dieu avec celle choisi par l'utilisateur, via un *radioGroup* par exemple.

Notre classe *BoucleDeJeu* est la boucle de jeu de notre application et c'est elle qui est exécutée dans le thread. A chaque tour de boucle, elle notifie ces *listener* (dieu par exemple) via la méthode *notifierListener().* Elle possède aussi des booléen, *enable* pour savoir si elle est activée ou non, pour lui dire de s'arrêter à la fermeture de l'application, et *played* pour mettre en pause le jeu.