**Description du diagramme de classe**

**Binôme :**

* **Salaheddine HASSNAOUI**
* **Yazid AKKI**

On a travaillé avec le modèle **MVC** pour faire la séparation entre les classes de données, métier et présentation.

Dans le package model la classe **Case** c’est une cellule de la grille qui va contenir dans notre exemple un bonbon, classe **GameModele** qui s’occupera de la génération de la grille, classe **BonbonModele** qui a un attribut type et une méthode pour créer des bonbons en utilisant la classe **BonbonFactory** ou cette dernière implémente l’interface **IBonbonFactory** on a utilisé le pattern **Factory** pour la génération des bonbons comme ca si on après on veut ajouter d’autre type il suffit d’ajouter juste une classe qui implémente cette interface.

Dans le package controler on a utilisé le pattern **strategy** on a un interface **Strategy** qui a la méthode repaint() après il y a deux classe qui implémente cet interface **Strategy2Echange** qui appel la méthode repait pour redessiner le jeux lorsqu’il y a un échange valide avec trois couleur et **Strategy3Echange** qui appel la méthode repaint pour redessiner le jeux lorsqu’il y a un échange valide avec quatre couleur, comme ca si jamais on veut ajouter des options par exemple si on a échange valide de n couleur on peut redéfinir la méthode pour mettre ce qu’on veut, utilisation du pattern **state** on a la classe **GameState** qui définit la méthode bonbonClicked() qui donnera un état au bonbon selon l’action de l’utilisateur cette dernière utilise la classe **ActionObeserveur** pour observer les actions de l’utilisateur, deux classe hérite de GameState on a **StateSelection** ou le bonbon a état par exemple 1 lorsqu’il est sélectionné et **StateSwap** ou le bonbon a état par exemple 2 lorsqu’il est sélectionné ensuite on a l’interface **IGame** qui définit les méthodes principales du jeux et qui implémenté par GameModel on a la classe **ActionObserver** qui définit les méthode qui notify selon l’action de l’utilisateur.

Dans le package view on a deux classes **ScorePanel** qui affiche le score et **GamePanel** qui affiche le jeu et qui a une association avec la classe **BonbonView** qui affiche les bonbons,

Classe **ActionObeserveur** pour observer les actions de l’utilisateur.

Pattern **Singleton** avec la classe joueur pour avoir une seule instance pour être sure qu’on a un joueur unique.