*JORET Eddy et CAZALIS Pauline*

*Rapport livrable 1 Developpement web avancé*

Table des matières

[I) Schéma de la base de données 2](#_Toc68170479)

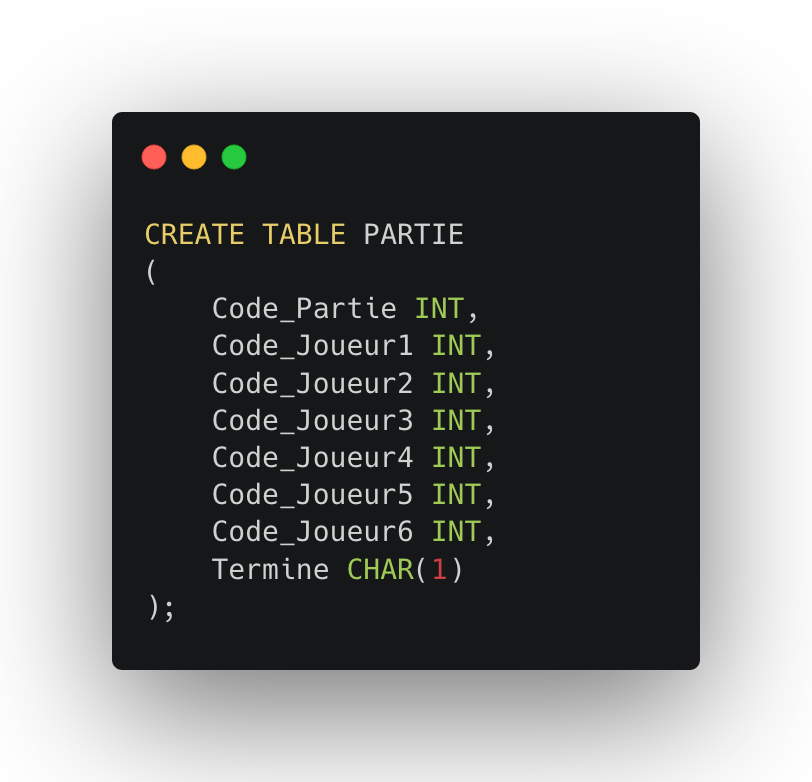
[II) Diagramme de classe UML des POJO 5](#_Toc68170480)

[III) Description des méthodes d’accès aux données 6](#_Toc68170481)

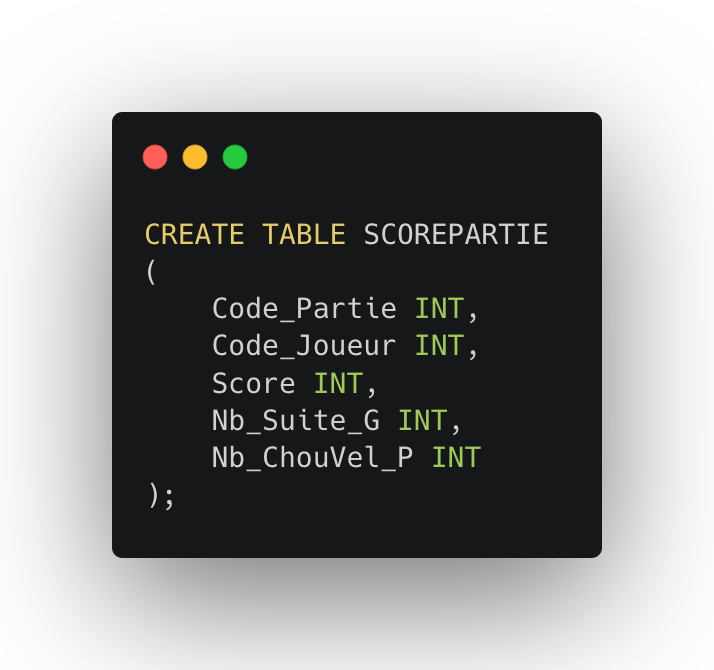
# Schéma de la base de données

Commandes SQL de la création des tables dans la base de données :

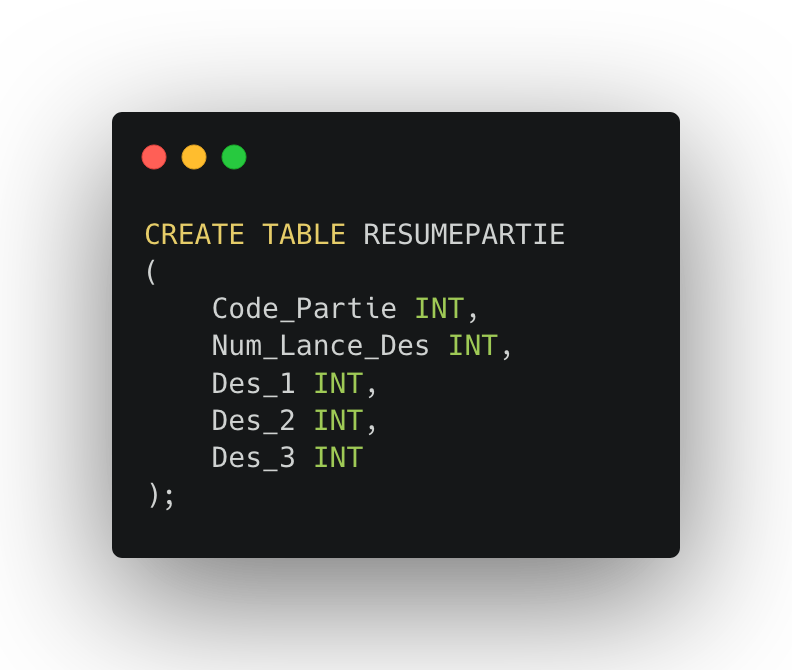
* Création de la table PARTIE



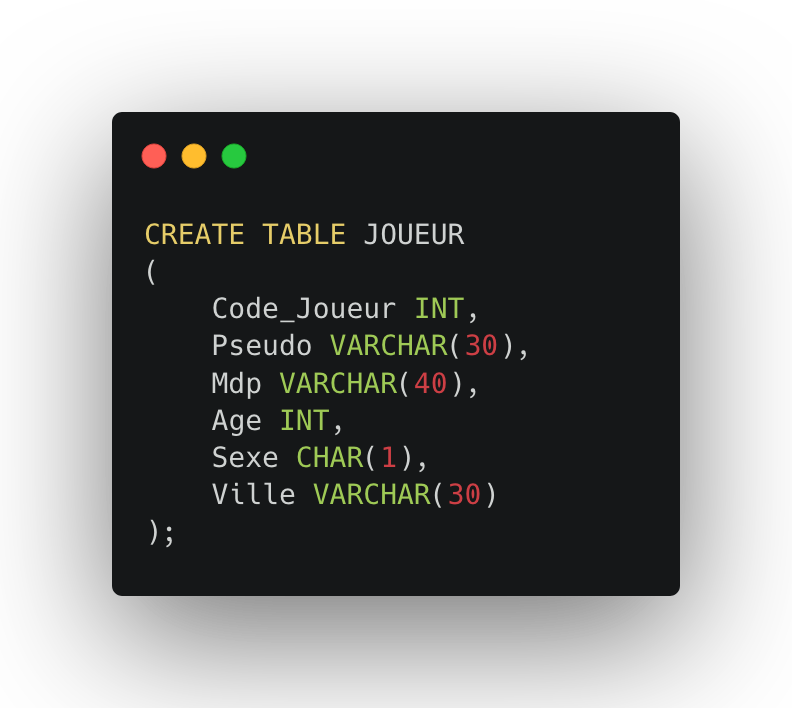
* Création de la table SCOREPARTIE



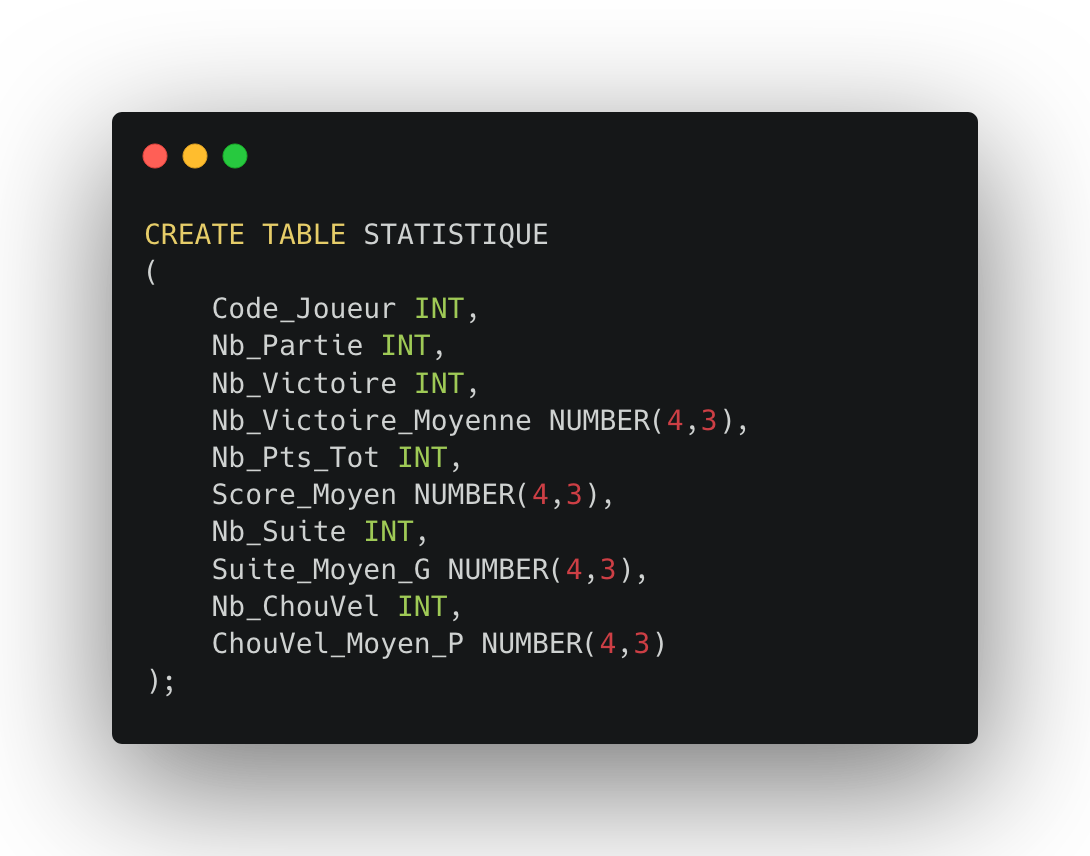
* Création de la table RESUMEPARTIE



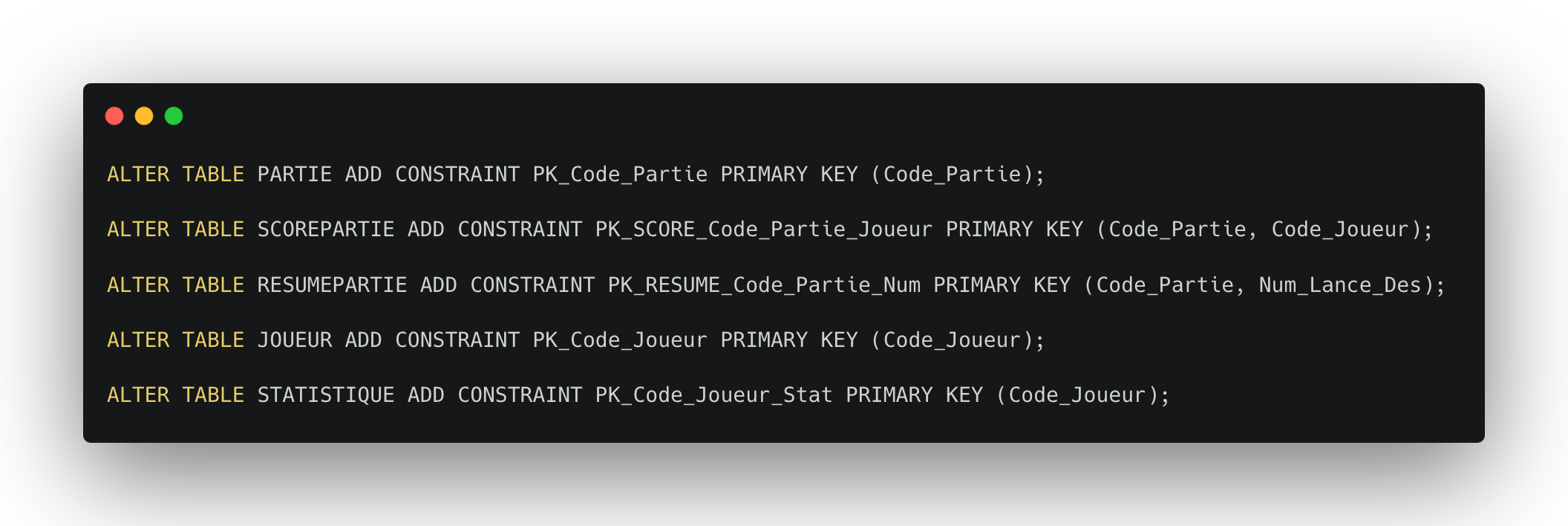
* Création de la table JOUEUR



* Création de la table STATISTIQUE



* Création des clés primaires



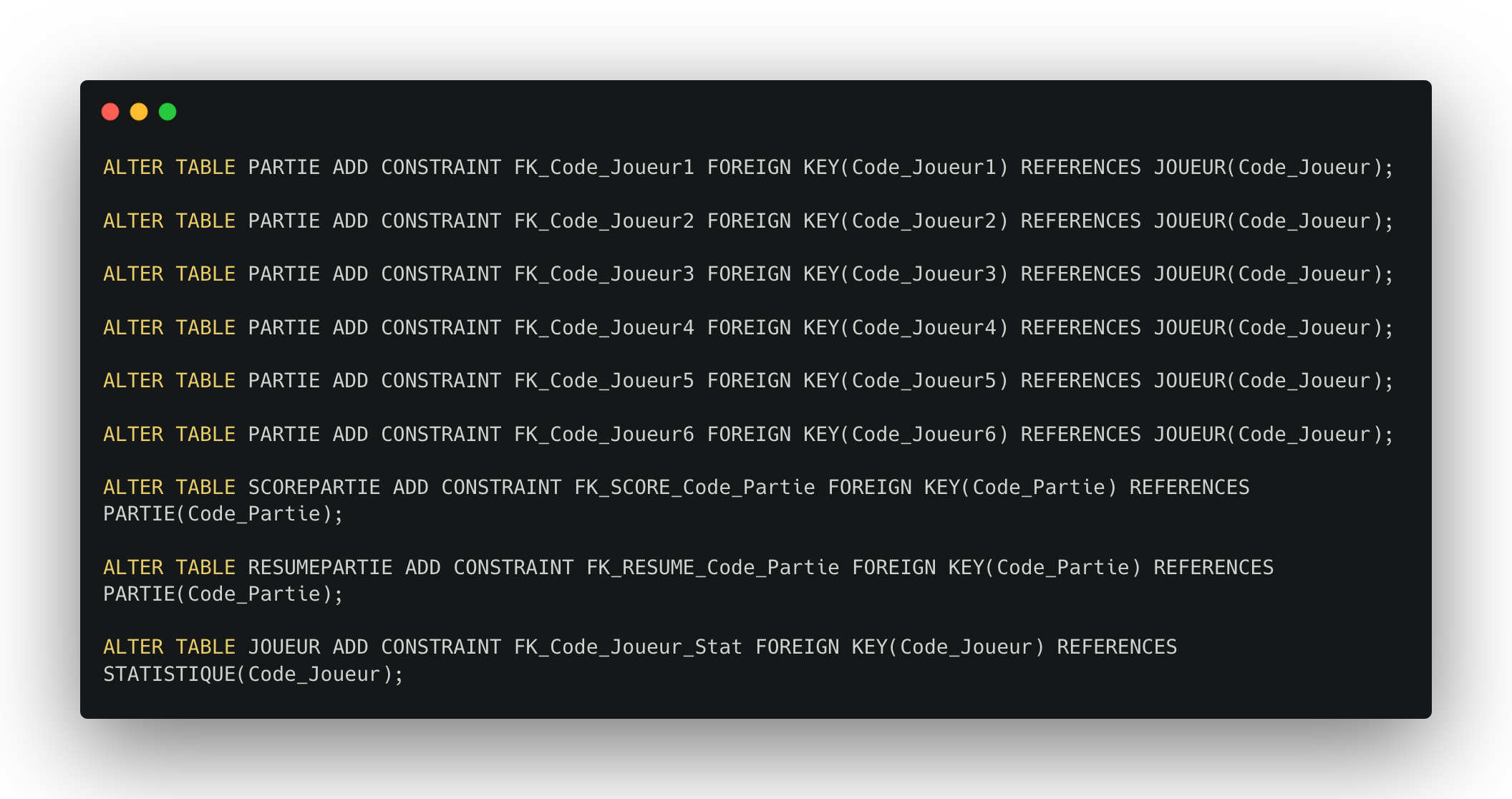
* Création des clés étrangères

Schéma UML de la base de données :

# Diagramme de classe UML des POJO

Après la création de notre base de données, nous avons pu générer automatiquement les POJO de notre base.

Ainsi, grâce au logiciel Netbeans, nous avons pu nous connecter à notre base de données SqlDevelopper, sélectionner nos tables et faire la génération de nos POJO.

Voici le diagramme de classe UML de nos POJO générés :

# Description des méthodes d’accès aux données

## FctScorePartie

Cette classe permet d’obtenir les différents scores de la partie comme par exemple la moyenne des suites gagnées ou encore le score total.

Ainsi nous avons créé plusieurs fonctions :

* InitScorePartie : cette fonction permet d’initialiser le code de la partie ainsi que le code du joueur pour obtenir son score.
* Pour tous les joueurs
* getScore\_Total : cette fonction permet de récupérer le score total d’un joueur lors d’une partie.
* getMoy\_Suite\_G : cette fonction permet de récupérer la moyenne du nombre de suite gagnée lors d’une partie.
* getMoy\_ChouVel\_P : cette fonction permet de récupérer la moyenne de Chouette Velute perdues lors d’une partie.
* Pour un joueur
* getValeurs : cette fonction permet l’affichage du score, du nombre de suite gagnées et du nombre du Chouette Velute perdue suivant le code de la partie et du code joueur. En effet, nous récupérons toutes les données en fonction du code partie et du code joueur, puis, suivant ces données, nous les affichons au format JSON.

La méthode executeQuery() permet d’interroger notre base de données pour ne rechercher que ce que l’on demande. Nous parcourons ensuite ces données une à une pour pouvoir les récupérer.

* getValeur : cette fonction permet l’affichage d’une donnée précise en fonction du code de la partie et du code du joueur. En effet, nous affichons au format JSON la valeur demandée par l’utilisateur en fonction du code de la partie et du code du joueur. Même procédé qu’avec la fonction getValeurs.
* Mise à jour des valeurs
* incScore : cette fonction permet l’incrémentation du score. A partir de la fonction getValeur, nous récupérons la colonne « score » qui va nous donner la valeur du score actuel et l’incrémenter avec le nouveau score obtenu. Ainsi, nous allons mettre à jour sa valeur grâce à un update correspondant au code de la partie et au code joueur demandé.
* incNb\_Suite\_G : cette fonction permet de mettre à jour la valeur du nombre de suite gagnée. Grâce à la fonction getValeur, nous pouvons récupérer la valeur de la colonne « suite » et l’incrémenter avec la nouvelle valeur. Ainsi, nous allons mettre à jour sa valeur grâce à un update correspondant au code de la partie et au code joueur demandé.
* incNb\_ChouVel\_P : cette fonction permet de mettre à jour le nombre de Chouette Velute perdue. Grâce à la fonction getValeur, nous pouvons récupéer la valeur de la colonne « chouvel » et l’incrémenter avec la nouvelle valeur. Ainsi, nous allons mettre à jour sa valeur grâce à un update correspondant au code de la partie et au code joueur demandé.

## FctResumePartie

Cette classe permet d’obtenir les informations concernant le déroulement de la partie comme par exemple le nombre total de lancer de dés.

Ainsi, nous avons créé plusieurs fonctions pour répondre à nos besoins :

* initTourResumePartie : cette fonction permet d’initialiser les valeurs d’une partie en fonction du code de la partie.
* Pour toutes les parties
* getTotal\_Lance : cette fonction permet d’obtenir le nombre total de lancer de dés fait en une partie.
* getTotal\_Nb\_Lance\_De : cette fonction permet d’obtenir le nombre total de fois qu’une valeur précise d’un des dés est tombé.
* getTotal\_Moyenne\_Lance\_De : cette fonction permet d’obtenir la moyenne totale d’une valeur de dés. Pour cela, nous utilisons la fonction getTotal\_Lance() et la fonction getTotal\_Nb\_Lance\_De() créés précédemment qui vont nous donner le nombre total de lancer de dés et le nombre total de fois qu’une valeur précise d’un des dés qui est tombé et nous allons les diviser ensemble pour nous donner la moyenne.
* Pour une partie
* getNb\_Lance : cette fonction permet d’obtenir le nombre de lancer de dés.
* getNb\_Lance\_De : cette fonction permet d’obtenir le nombre de fois qu’une valeur précise d’un des dés est tombée.
* getMoyenne\_Lance\_De : cette fonction permet d’obtenir la moyenne d’une valeur de dés. En effet, nous allons obtenir ce résultat en divisant le nombre de lancer de dés en une partie et le nombre de fois qu’une valeur précise d’un dés est tombé grâce au deux fonctions que nous avons créés précédemment (getNb\_Lance() et getNb\_Lance\_De() ).
* getDerniers\_Des : cette fonction permet d’obtenir les 3 dernières valeurs du dernier lancer de dés. Nous stockons ces valeurs dans un tableau de int.
* Mise à jour
* majDes : cette fonction permet de mettre à jour la valeur des 3 dés à la fin du tour d’un joueur à l’aide du code de la partie et du tableau qui contient les valeurs des dés, stocké précédemment. Pour cela, nous allons récupérer le nombre de lancer de dés pour une partie grâce à la fonction getNb\_Lance(). Une mise à jour de ces valeurs sera effectuée grâce à la fonction update.

## FctPartie

Cette classe permet d’obtenir les informations