*JORET Eddy et CAZALIS Pauline*

*Rapport livrable 1 Developpement web avancé*

Table des matières

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

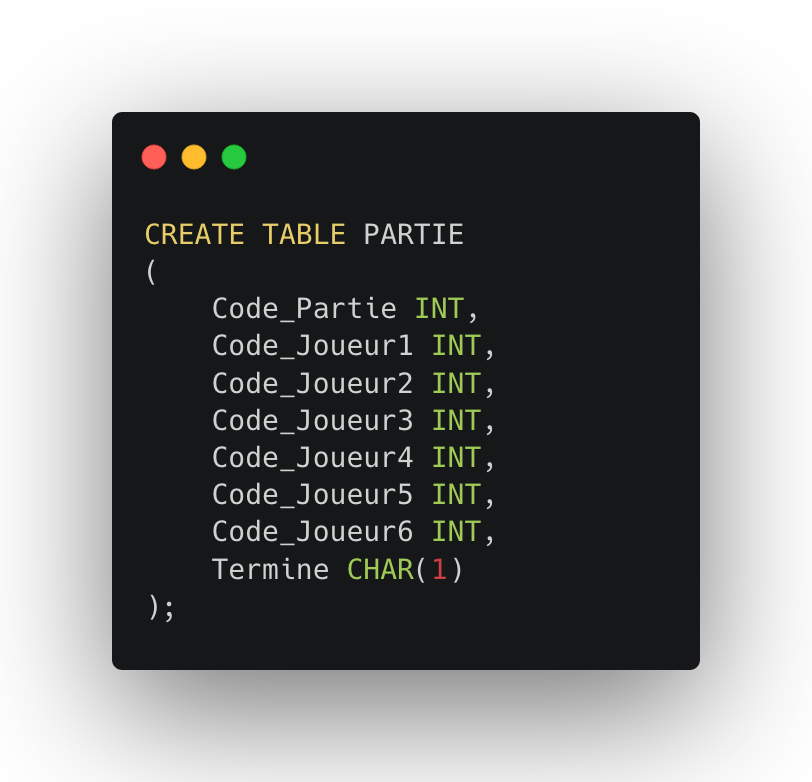
[II) Diagramme de classe UML des POJO 5](#_Toc68170480)

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

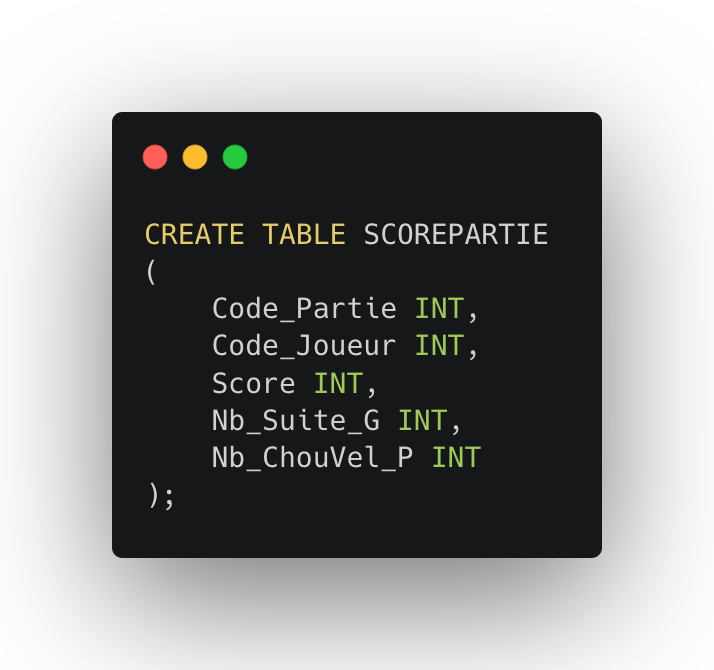
# Schéma de la base de données

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

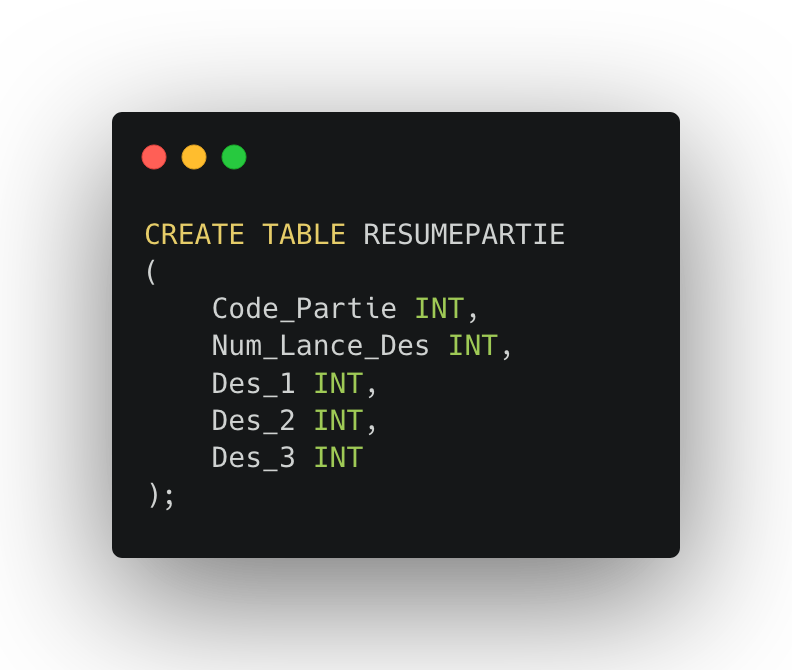
* Création de la table PARTIE



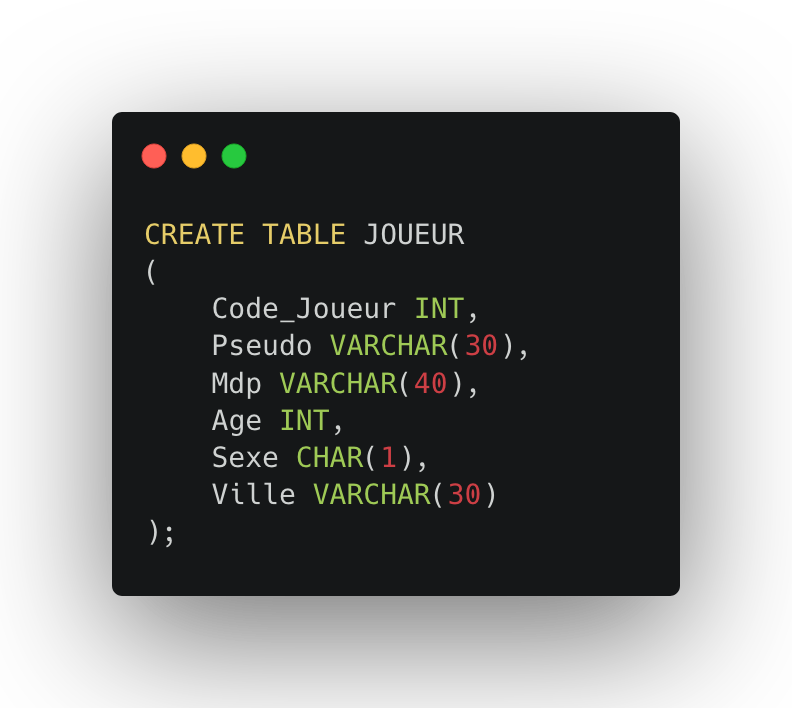
* Création de la table SCOREPARTIE



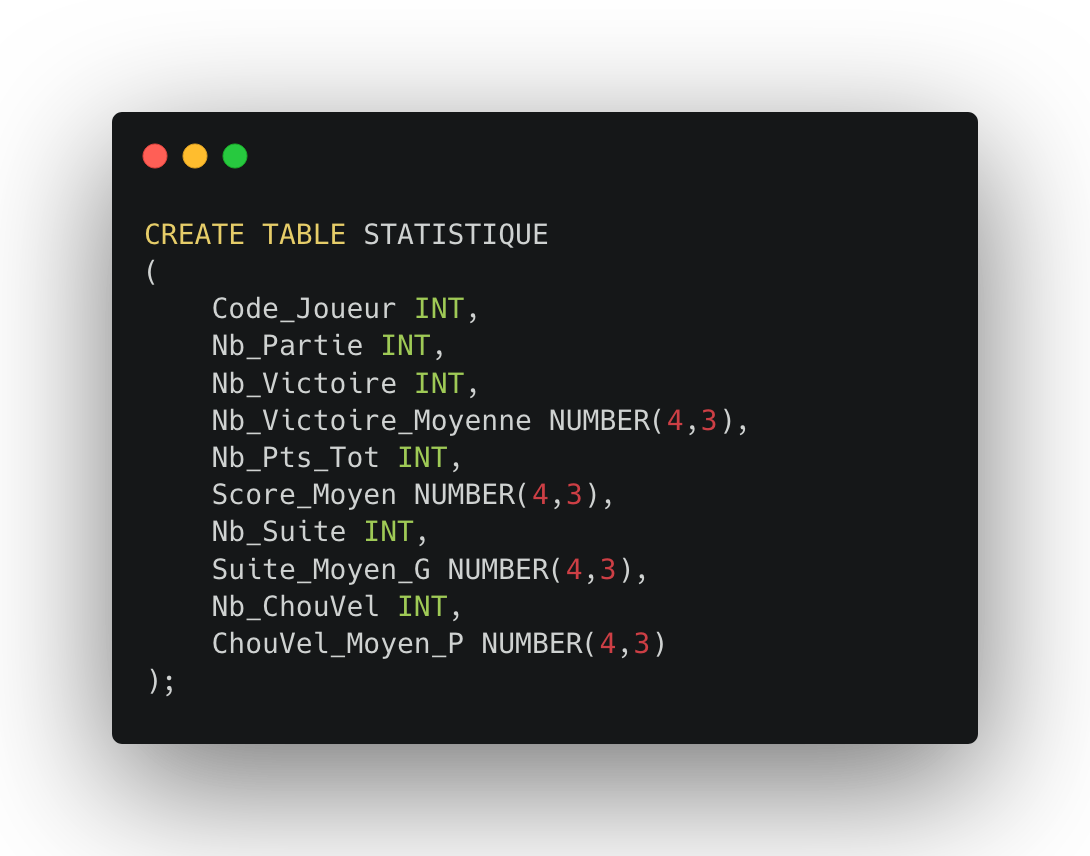
* Création de la table RESUMEPARTIE



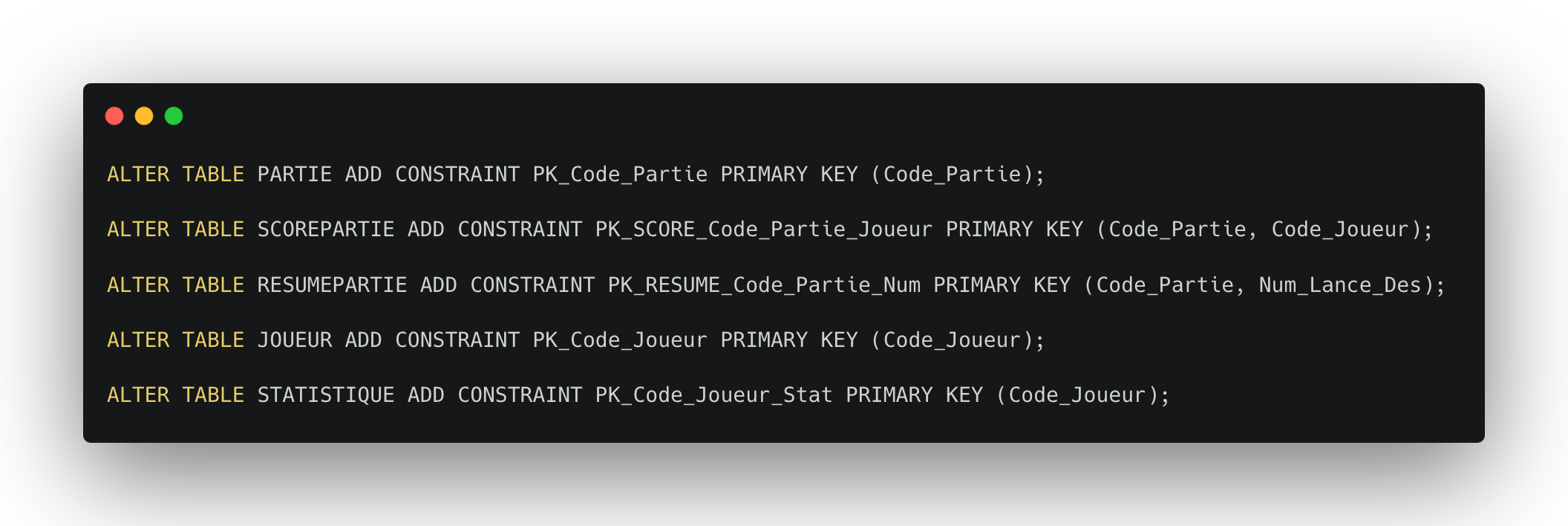
* Création de la table JOUEUR



* Création de la table STATISTIQUE



* Création des clés primaires



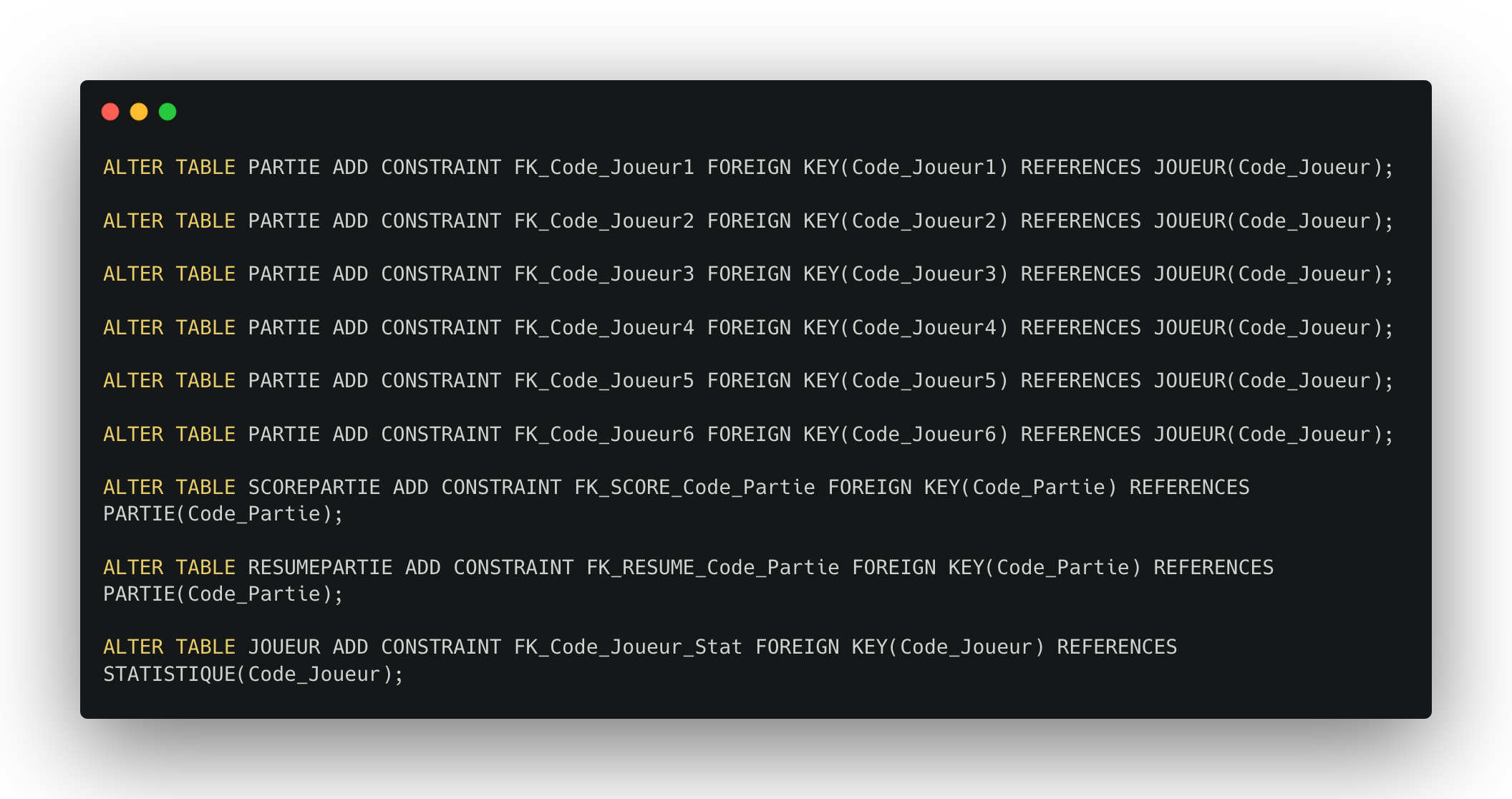
* Création des clés étrangères

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 d’une 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 celles-ci, 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 et les afficher.

* 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 du joueur. 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 récupérer 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 lors d’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 lors d’une partie.
* 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 est tombé et nous allons les diviser ensemble pour nous donner la moyenne attendue.
* Pour une partie
* getNb\_Lance : cette fonction permet d’obtenir le nombre de lancer de dés correspondant à une partie dont le code est passé en paramètre.
* getNb\_Lance\_De : cette fonction permet d’obtenir, pour une partie dont le code est passé en paramètre, 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 pour une partie dont le code est passé en paramètre. 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 aux 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 lors d’une partie dont le code est passé en paramètre. Nous stockons ces valeurs dans un tableau de int et les affichons.
* 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 sur une partie en cours comme par exemple le nombre de joueurs présent dans une partie.

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

* initPartie() : cette fonction permet d’initialiser, avec le tableau contenant les codes joueur en paramètre, une partie. En effet, nous récupérons le nombre de parties totales faites et nous lui ajoutons la nouvelle que le joueur va lancer. À partir de ce moment-là, nous créons un code pour la partie et nous initialisons nos paramètres grâce à la fonction initScorePartie qui appartient à la classe FctScore.
* Pour toutes les parties
* getNb\_Partie\_Tot : cette fonction permet de compter le nombre de partie totale faites.
* getNb\_Partie\_Tot\_En\_Cours : cette fonction permet de compter le nombre de partie totale en cours.
* getNb\_Joueur\_Tot : cette fonction permet d’obtenir le nombre de joueur total. Le nombre de joueur pour chaque partie est limité à 6. Ainsi nous ne regardons que s’il y a au moins deux personnes qui joue et si le nombre est inférieur à 6.
* getMoy\_Joueur : cette fonction permet d’obtenir la moyenne du nombre de joueur par partie. Pour cela, nous récupérons le nombre de partie total (getNb\_Partie\_Tot) et le nombre de joueur total (getNb\_Joueur\_Tot) et nous les divisons ensembles pour obtenir la moyenne.
* getMoy\_Score\_Partie\_F : cette fonction permet de calculer le score moyen des parties terminées. En effet, nous récupérons les données de la partie si celle-ci n’est pas terminée, puis, nous regardons si le nombre de joueur est inférieur à 6 et s’il n’est pas nul et nous recherchons pour chaque joueur et partie les valeurs dans la colonne score. Enfin, nous divisons ces valeurs obtenues par le nombre de partie pour obtenir notre résultat.
* Pour une partie
* getNb\_Joueur : cette fonction permet d’obtenir le nombre de joueur pour une partie définie en paramètre.
* getJoueur : cette fonction permet d’obtenir la liste des joueurs en fonction du code de la partie passé en paramètre. Nous conservons ces valeurs dans un tableau de string. Pour obtenir la liste des joueurs, nous sélectionnons toutes les valeurs de la partie suivant son code, puis nous recherchons les pseudos de chaque joueur grâce à la fonction getPseudo de la classe FctJoueur.
* getFinish : cette fonction permet de savoir si la partie est terminée ou non.
* getStats\_Joueurs : cette fonction permet d’obtenir, pour chaque partie, les statistiques de chaque joueur suivant un code partie défini en paramètre de la fonction. Nous conservons ces données dans un tableau de string. Nous obtenons le pseudo du joueur grâce à la fonction getPseudo de la classe FctJoueur, puis nous affichons les statistiques de ce joueur grâce à la fonction getStats\_Pseudo avec en paramètre le pseudo récupéré avant.
* getStats\_Joueur : cette fonction permet d’obtenir les statistiques d’un joueur pour une partie en fonction de son pseudo. Nous obtenons ses statistiques grâce à la fonction getStats\_Pseudo de la classe FctJoueur.
* getStat\_Joueur : cette fonction permet d’obtenir une statistique précise du joueur pour une partie. Nous obtenons cette valeur grâce à la fonction getStat\_Pseudo de la classe joueur qui récupère, en fonction du pseudo passé en paramètre et de la colonne demandée, la valeur demandée.
* Mise à jour
* majScore : cette fonction permet de mettre à jour le score du joueur en fonction du code de la partie, du pseudo et du score actuel. Dans un premier temps, nous obtenons le code du joueur en fonction du pseudo passé en paramètre grâce à la fonction getCode\_Joueur de la classe FctJoueur, puis, nous incrémentons le score grâce à la fonction incScore de la classe FctJoueur.
* majSuite\_G : cette fonction permet de mettre à jour le nombre de suite gagnée du joueur en fonction du code de la partie, du pseudo et du nombre de suite gagnée actuel. Dans un premier temps, nous obtenons le code du joueur en fonction du pseudo passé en paramètre grâce à la fonction getCode\_Joueur de la classe FctJoueur, puis, nous incrémentons ce nombre grâce à la fonction incNb\_Suite\_G de la classe FctJoueur.
* majChouVel\_P: cette fonction permet de mettre à jour le nombre de Chouette Velute perdue du joueur en fonction du code de la partie, du pseudo et du nombre de Chouette velute perdue actuel. Dans un premier temps, nous obtenons le code du joueur en fonction du pseudo passé en paramètre grâce à la fonction getCode\_Joueur de la classe FctJoueur, puis, nous incrémentons ce nombre grâce à la fonction incNb\_ChouVel\_P de la classe FctJoueur.
* initLanc : cette fonction permet d’enregistrer le dernier lancer de dés effectué dans une partie. Nous récupérons la dernière valeurs grâce à la fonction initTourResumePartie de la classe FctResume, puis nous mettons à jour celle-ci grâce à la fonction majDes de la classe FctResume.

## FctJoueur

Cette classe permet d’obtenir les informations sur le ou les joueurs comme par exemple son pseudo ou encore la moyenne d’âge des joueurs jouant au jeu.

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

* InitJoueur : cette fonction permet d’initialiser les champs pour la création du joueur. Ainsi nous insérons dans notre table joueur, les données passées en paramètre tel que le pseudo, le mode de passe, l’âge, le sexe et la ville du joueur.
* Pour tous les joueurs
* getNb\_Tot\_Joueur : cette fonction permet d’obtenir le nombre total de joueur
* getListe\_Pseudo : cette fonction permet d’obtenir la liste des pseudos des joueurs. Ils sont stockés dans un tableau de string.
* getMoy\_Age : cette fonction permet de calculer la moyenne des âges des joueurs.
* getMoy\_Sexe : cette fonction permet de calculer la moyenne d’homme et de femme qui joue au jeu.
* getListe\_Villes : cette fonction permet d’obtenir la liste des villes des différents joueurs. Les données sont stockées dans un tableau de string.
* Pour un joueur
* getPseudo : cette fonction permet d’obtenir le pseudo d’un joueur à partir du code du joueur. Pour cela, nous recherchons les données correspondant au code joueur dans la classe Joueur grâce à la méthode find, puis nous demandons son pseudo grâce à la méthode getPseudo.
* getCode\_Joueur : cette fonction permet d’obtenir le code joueur à partir du pseudo du joueur. Pour cela, nous recherchons les données correspondant au pseudo du joueur dans la classe Joueur grâce à la methode find, puis nous demandons le code joueur grâce à la méthode getCodeJoueur.
* getInfos\_Pseudo : cette fonction permet d’obtenir les informations du joueur à partir du pseudo de celui-ci passé en paramètre. Nous récupérons nos données tels que l’âge, le sexe et la ville du joueur grâce à la méthode find qui va récupérer de notre classe Joueur les informations à partir du pseudo donné. De ce fait, nous obtenons nos données en appelant les différentes méthodes correspondantes et nous les affichons dans un string.
* getStats\_Pseudo : cette fonction permet d’obtenir les statistiques d’un joueur à partir de son pseudo. Nous obtenons celles-ci en appelant la fonction getStats de notre classe FctStat.
* getInfo\_Pseudo : cette fonction permet d’obtenir une information du joueur à partir de son pseudo. Nous récupérons d’abord les données de notre classe Joueur à partir de son pseudo grâce à la méthode find, puis, nous regardons suivant la colonne demandée l’information voulue.
* getStat\_Pseudo : cette fonction permet d’obtenir une statistique du joueur à partir de son pseudo. Nous obtenons celle-ci en appelant la fonction getStat de notre classe FctStat.
* Mise à jour
* majMdp : cette fonction permet la mise à jour du mot de passe du joueur. Pour cela on recherche les informations à partir du pseudo de celui-ci dans notre classe Joueur, puis nous appelons la fonction setMdp avec le mot de passe passé en paramètre.
* majAge : cette fonction permet la mise à jour de l’âge du joueur. Pour cela on recherche les informations à partir du pseudo de celui-ci dans notre classe Joueur, puis nous appelons la fonction setAge avec l’âge passé en paramètre.
* majSexe : cette fonction permet la mise à jour du sexe du joueur. Pour cela on recherche les informations à partir du pseudo de celui-ci dans notre classe Joueur, puis nous appelons la fonction setSexe avec le sexe passé en paramètre.
* majVille : cette fonction permet la mise à jour de la ville du joueur. Pour cela on recherche les informations à partir du pseudo de celui-ci dans notre classe Joueur, puis nous appelons la fonction setVille avec la ville passée en paramètre.

## FctStatistique

Cette classe permet d’obtenir les statistiques d’un joueur comme par exemple le nombre de points total qu’il a gagné.

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

* InitStat : cette fonction permet d’initialiser les statistiques d’un joueur à partir de son code joueur.
* Pour tous les joueurs
* getMoy\_Partie\_Tot : cette fonction permet de calculer la moyenne de partie jouée.
* getMoy\_Partie\_G\_Tot : cette fonction permet de calculer la moyenne de partie gagnée.
* getNb\_Pts\_Tot : cette fonction permet d’obtenir le nombre de points total gagné.
* getScore\_Moy\_Tot : cette fonction permet de calculer la moyenne du score moyen.
* getSuite\_Moy\_G\_Tot : cette fonction permet de calculer la moyenne du nombre moyen de suites gagnées.
* getChouVel\_P\_Tot : cette fonction permet de calculer la moyenne du nombre moyen de Chouette Velute perdue.
* Pour un joueur
* getStats : cette fonction permet d’obtenir les statistiques d’un joueur. Pour obtenir ce résultat, nous recherchons dans la classe statistique les données correspondant au code joueur. A partir de ceci, nous affichons les statistiques de celui-ci grâce à la fonction correspondante. Les résultats sont affichés dans un string.
* getStat : cette fonction permet d’afficher une statistique particulière par rapport à un code joueur passé en paramètre. Pour cela, nous recherchons les données correspondant au code joueur, puis, nous affichons le résultat demandé. Le résultat est affiché dans un string.
* Mise à jour
* majStats : cette fonction permet de mettre à jour toutes les statistiques d’un joueur c’est-à-dire le nombre de victoire moyenne, le score moyen, le nombre de suites moyennes gagnées ou encore le nombre de Chouette Velute moyen perdue. Pour cela nous appelons les fonctions correspondantes aux différentes mises à jour.
* incPartie : cette fonction permet d’incrémenter le nombre de partie du joueur. Pour cela, nous recherchons dans la classe Statistique, les informations correspondant au code joueur, passé en paramètre, puis, nous appelons la fonction setNbPartie qui va nous permettre de mettre à jour ce nombre.
* incVictoire : cette fonction permet d’incrémenter le nombre de victoire du joueur. Pour cela, nous recherchons dans la classe Statistique les informations correpondant au code joueur, passé en paramètre, puis, nous appelons la fonction setNbVictoire qui va nous permettre de mettre à jour ce nombre.
* majVictoire\_Moyenne : cette fonction permet de mettre à jour le nombre de victoire moyen du joueur. Pour cela, nous regardons si le joueur a gagné. Si c’est le cas, nous incrémentons le nombre de victoire grâce à la fonction incVictoire, puis, nous recherchons dans la classe Statistique toutes les informations correspondant au code joueur, passé en paramètre. Enfin, nous appelons la fonction setNbVictoireMoyenne qui va nous permettre de mettre à jour ce nombre.
* incPts : cette fonction permet d’incrémenter le nombre de point du joueur. Pour cela, nous recherchons dans la classe Statistique les informations correspondant au code joueur, passé en paramètre, puis, nous appelons la fonction setNbPtsTot qui va nous permettre de mettre à jour ce nombre.
* majScore\_Moyen : cette fonction permet de mettre à jour le score moyen d’un joueur. Pour cela, nous appelons la fonction incPts, qui incrémente le nombre de points, puis, nous recherchons dans la classe Statistique les informations correspondant au code joueur et nous mettons à jour le score moyen grâce à la fonction setScoreMoyen.
* incSuite : cette fonction permet d’incrémenter le nombre de suite gagnée du joueur. Pour cela, nous recherchons dans la classe Statistique les informations correspondant au code joueur, passé en paramètre, puis, nous appelons la fonction setNbSuite qui va nous permettre de mettre à jour ce nombre.
* majSuite\_Moyen\_G : cette fonction permet de mettre à jour le nombre de suite moyenne gagnée par un joueur. Pour cela nous recherchons les informations correspondant au code joueur, passé en paramètre, dans la classe Statistique. Puis, nous multiplions le nombre de suite (getNbSuite) avec le nombre de suite moyen gagné (getSuiteMoyenG). Nous ajoutons à cette valeur, le nombre de suite passé en paramètre. Nous incrémentons le nombre de suite gagnée grâce à la fonction incSuite. Enfin, nous obtenons notre résultat final en appelant la fonction setSuiteMoyenG, avec pour paramètre notre moyenne de suite gagnée, obtenu précédemment.
* incChouVel : cette fonction permet d’incrémenter le nombre de Chouette Velute perdue du joueur. Pour cela, nous recherchons dans la classe Statistique les informations correspondant au code joueur, passé en paramètre, puis, nous appelons la fonction setNbChouVel qui va nous permettre de mettre à jour ce nombre.
* majChouVel\_Moyen\_P : cette fonction permet de mettre à jour le nombre moyen de Chouette Velute perdue par un joueur. Pour cela nous recherchons les informations correspondant au code joueur, passé en paramètre, dans la classe Statistique. Puis, nous multiplions le nombre de Chouette Velute (getNbChouVel) avec le nombre de Chouette Velute moyen perdue (getChouvelMoyenP). Nous ajoutons à cette valeur, le nombre de Chouette Velute perdue passé en paramètre. Nous incrémentons le nombre de Chouette Velute perdue grâce à la fonction incChouVel. Enfin, nous obtenons notre résultat final en appelant la fonction setChouVelMoyenP, avec pour paramètre notre moyenne de Chouette Velute perdue, obtenue précédemment.