RAPPORT LIVRABLE 1 DEVELOPPEMENT WEB AVANCE

JORET Eddy et CAZALIS Pauline

Table des matières

I) Schéma de la base de données

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

➔ Création de la table PARTIE

```sql
CREATE TABLE PARTIE
(
    Code_Partie INT,
    Code_Joueur1 INT,
    Code_Joueur2 INT,
    Code_Joueur3 INT,
    Code_Joueur4 INT,
    Code_Joueur5 INT,
    Code_Joueur6 INT,
    Termine CHAR(1)
);
```

➔ Création de la table SCOREPARTIE

```sql
CREATE TABLE SCOREPARTIE
(
    Code_Partie INT,
    Code_Joueur INT,
    Score INT,
    Nb_Suite_G INT,
    Nb_ChouVel_P INT
);
```

➔ Création de la table RESUMEPARTIE

```sql
CREATE TABLE RESUMEPARTIE
(
    Code_Partie INT,
    Num_Lance_Des INT,
    Des_1 INT,
    Des_2 INT,
    Des_3 INT
);
```

➔ Création de la table JOUEUR

```sql
CREATE TABLE JOUEUR
(
    Code_Joueur INT,
    Pseudo VARCHAR(30),
    Mdp VARCHAR(40),
    Age INT,
    Sexe CHAR(1),
    Ville VARCHAR(30)
);
```

➔ Création de la table STATISTIQUE

```sql
CREATE TABLE STATISTIQUE
(
    Code_Joueur INT,
    Nb_Partie INT,
    Nb_Victoire INT,
    Nb_Victoire_Moyenne NUMBER(4,3),
    Nb_Pts_Tot INT,
    Score_Moyen NUMBER(4,3),
    Nb_Suite INT,
    Suite_Moyen_G NUMBER(4,3),
    Nb_ChouVel INT,
    ChouVel_Moyen_P NUMBER(4,3)
);
```

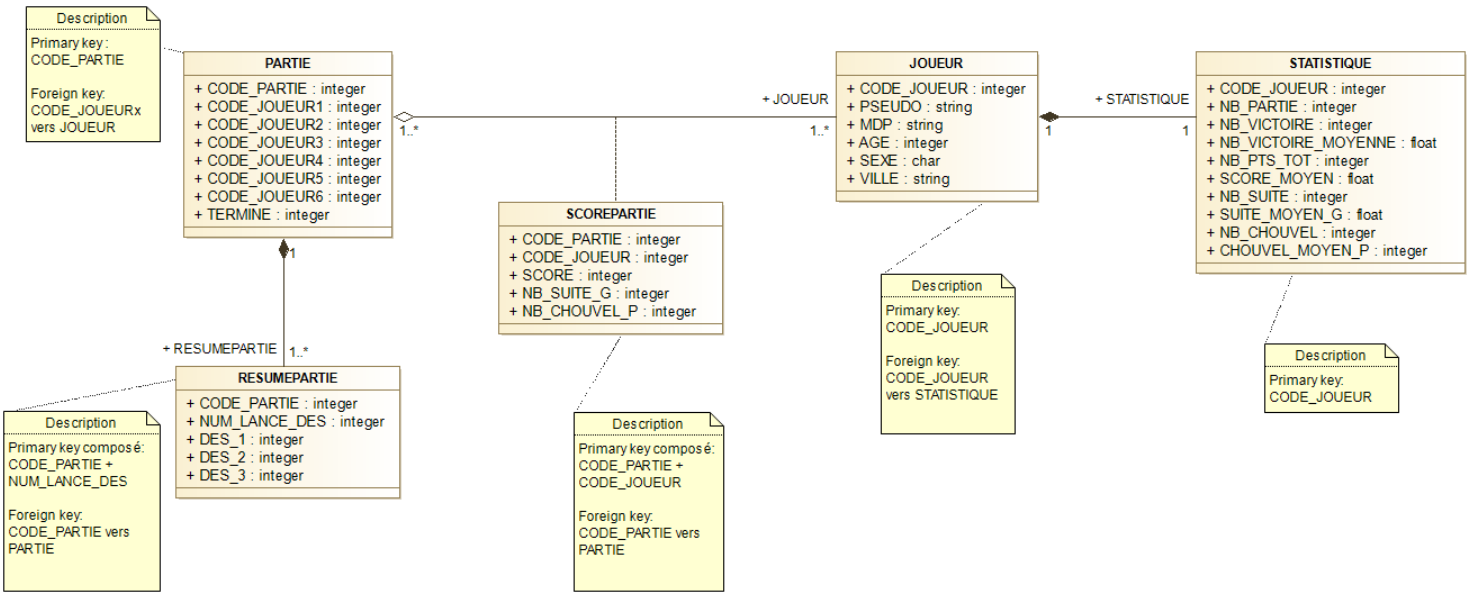
➔ Création des clés primaires

```sql
ALTER TABLE PARTIE ADD CONSTRAINT PK_Code_Partie PRIMARY KEY (Code_Partie);

ALTER TABLE SCOREPARTIE ADD CONSTRAINT PK_SCORE_Code_Partie_Joueur PRIMARY KEY (Code_Partie, Code_Joueur);

ALTER TABLE RESUMEPARTIE ADD CONSTRAINT PK_RESUME_Code_Partie_Num PRIMARY KEY (Code_Partie, Num_Lance_Des);

ALTER TABLE JOUEUR ADD CONSTRAINT PK_Code_Joueur PRIMARY KEY (Code_Joueur);

ALTER TABLE STATISTIQUE ADD CONSTRAINT PK_Code_Joueur_Stat PRIMARY KEY (Code_Joueur);
```

➔ Création des clés étrangères

```
ALTER TABLE PARTIE ADD CONSTRAINT FK_Code_Joueur1 FOREIGN KEY(Code_Joueur1) REFERENCES JOUEUR(Code_Joueur);

ALTER TABLE PARTIE ADD CONSTRAINT FK_Code_Joueur2 FOREIGN KEY(Code_Joueur2) REFERENCES JOUEUR(Code_Joueur);

ALTER TABLE PARTIE ADD CONSTRAINT FK_Code_Joueur3 FOREIGN KEY(Code_Joueur3) REFERENCES JOUEUR(Code_Joueur);

ALTER TABLE PARTIE ADD CONSTRAINT FK_Code_Joueur4 FOREIGN KEY(Code_Joueur4) REFERENCES JOUEUR(Code_Joueur);

ALTER TABLE PARTIE ADD CONSTRAINT FK_Code_Joueur5 FOREIGN KEY(Code_Joueur5) REFERENCES JOUEUR(Code_Joueur);

ALTER TABLE PARTIE ADD CONSTRAINT FK_Code_Joueur6 FOREIGN KEY(Code_Joueur6) REFERENCES JOUEUR(Code_Joueur);

ALTER TABLE SCOREPARTIE ADD CONSTRAINT FK_SCORE_Code_Partie FOREIGN KEY(Code_Partie) REFERENCES
PARTIE(Code_Partie);

ALTER TABLE RESUMEPARTIE ADD CONSTRAINT FK_RESUME_Code_Partie FOREIGN KEY(Code_Partie) REFERENCES
PARTIE(Code_Partie);

ALTER TABLE JOUEUR ADD CONSTRAINT FK_Code_Joueur_Stat FOREIGN KEY(Code_Joueur) REFERENCES
STATISTIQUE(Code_Joueur);
```

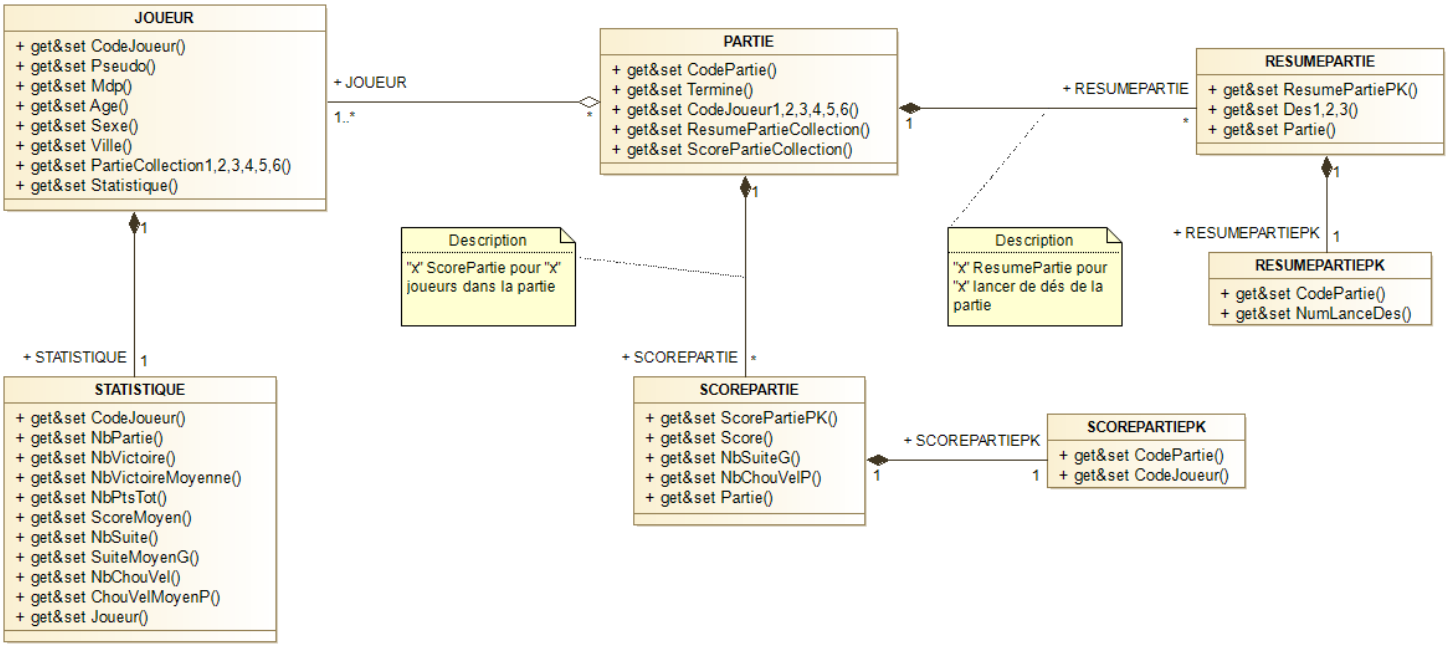
Schéma UML de la base de données :



II) 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 :



III) Description des méthodes d'accès aux données

1) 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 la ligne correspondante pour garder le score, le nombre de suite gagnée et de Chouette Velute perdue à partir du code de la partie ainsi que du code du joueur.

➔ Pour tous les joueurs

- getScore_Total : cette fonction permet de récupérer le score total 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 de récupérer le score, le nombre de suite gagnées et le nombre du Chouette Velute perdue suivant le code de la partie et du code joueur en les mettant au format JSON.

- getValeur : cette fonction permet de récupérer une donnée précisée en fonction du code de la partie et du code du joueur.

➔ 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 « NB_SUITE_G » 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 « NB_CHOUVEL_P » 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.

2) 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 et en précisant automatiquement le numéro de lancer de cette partie.

➔ Pour toutes les parties

- getTotal_Lance : cette fonction permet d'obtenir le nombre total de lancer de dés fait lors de toutes les parties.

- getTotal_Nb_Lance_De : cette fonction permet d'obtenir le nombre total de fois qu'une valeur précise d'un des dés est tombée lors de toutes les parties.

- 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 d'Int.

- getDes : cette fonction permet d'obtenir les 3 valeurs des dès d'un lancer précis passé en paramètre avec le code de le partie correspondante.

- getScore_Des : cette fonction permet d'obtenir le nombre de points pour le dernier lancer de dés d'une partie.

➔ 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() pour savoir le nombre de lancer sur cette partie. Une mise à jour de ces valeurs sera effectuée grâce à la fonction update.

➔ Fonctions facultatives pour le calcul du score

- suite : cette fonction permet de savoir s'il y a une suite entre le dernier lancé de dés et celui juste avant qui sont passé en paramètres et en appelant les deux fonctions de vérification pour les deux types de suites possibles.

- suite1 : cette fonction permet de savoir s'il y a une suite de 1-2-3 de l'avant dernier lancé avec le dernier lancé qui sera de 3-4-5.

- suite2 : cette fonction permet de savoir s'il y a une suite de 2-3-4 de l'avant dernier lancé avec le dernier lancé qui sera de 4-5-6.

- culDeChouette : cette fonction permet d'obtenir le nombre de points réalisé avec le type de valeur du dé.

3) 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_Participation_Tot : cette fonction permet d'obtenir le nombre de participation total aux parties.

- 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_Participation_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. Pour cela nous allons, pour chaque joueur de ces parties, faire le total du score puis le diviser par le nombre de partie terminée.

➔ 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 une 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 reçus via les points du lancer. 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.

- incSuite_G : cette fonction permet d'incrémenter le nombre de suite gagnée du joueur en fonction du code de la partie et du pseudo. 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 de 1 grâce à la fonction incNb_Suite_G de la classe FctJoueur.

- incChouVel_P: cette fonction permet d'incrémenter le nombre de Chouette Velute perdue du joueur en fonction du code de la partie et du pseudo. 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 de 1 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 initialisons le dernier numéro de lancer de la partie 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.

4) 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, la ville du joueur et le code du joueur qui est créé automatiquement.

➔ 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 d'âge 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.

5) 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 totale.

- getMoy_Partie_G_Tot : cette fonction permet de calculer la moyenne de partie gagnée totale.

- 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 total.

- getSuite_Moy_G_Tot : cette fonction permet de calculer la moyenne du nombre de suites gagnées totale.

- getChouVel_P_Tot : cette fonction permet de calculer la moyenne du nombre de Chouette Velute perdue totale.

➔ 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'obtenir une statistique particulière par rapport à un code joueur passé en paramètre. Pour cela, nous recherchons les données correspondant au code joueur, puis, nous obtenons le résultat demandé. Le résultat est stocké via 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 correspondant 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.