Le proviseur d'un collège souhaite moderniser son système informatique en proposant aux parents d'élèves un suivi des notes de leurs enfants qu'ils obtiennent au cours de l'année scolaire.

Un sondage a montré que la totalité des parents d'élèves disposait d'une connexion internet à la maison ce qui autorise de proposer ce service à travers la mise en ligne d'une application web appelée **EvalNotes**.

Le principe retenu est une initialisation des données (classes, élèves, professeurs, etc.) avant la rentrée scolaire par l'informaticien du collège et un effacement total des données à la fin de l'année scolaire. Ainsi, aucune historisation des données des années précédentes n'est effectuée.

Dans notre cas d'étude, nous considérons que l'année scolaire en cours correspond au 01/09/2015 jusqu'au 30/06/2016.

Les utilisateurs du système

Les parents : Ce sont les parents des élèves (et non les élèves eux-mêmes) qui ont accès à certaines fonctionnalités telle que l'édition du bulletin de notes trimestriel. Ils peuvent avoir un ou plusieurs enfants inscrits dans l'établissement scolaire mais disposent d'un seul login pour cela.

Les professeurs : Ils alimentent le système en données en créant des devoirs et des notes pour les élèves. Ils peuvent aussi afficher quelques états donnant des statistiques sur les notes. Un devoir est donné par un professeur pour une classe et contient un titre de devoir. Ensuite le professeur crée autant de notes qu'il y a d'élèves dans la classe. Un professeur ne peut enseigner qu'une seule matière.

L'Administrateur : Deux personnes dans l'établissement ont ce rôle : l'informaticien salarié à temps partiel de l'établissement ainsi que le proviseur. Une semaine avant la rentrée scolaire, l'administrateur réinitialise la base de données en créant tous les utilisateurs et en effaçant toutes les données se rapportant aux notes de l'année scolaire précédente. Il peut aussi afficher quelques états donnant des statistiques sur les notes et gère le système des tutorats (ceci sera expliqué plus loin dans ce document).

Authentification

Afin de paramétrer le système, un courrier a été envoyé à tous les parents d'élèves avant la rentrée scolaire présentant succinctement les fonctionnalités du système et en leur attribuant à chacun un login et un mot de passe.

Par défaut, tous les mots de passe de tous les utilisateurs ont été initialisés avec la même chaîne de caractères : « *init* ».

Pour s'authentifier, chaque utilisateur (parent, professeur, administrateur) doit fournir sur la page d'accueil du site un login et un mot de passe corrects. Lors du premier accès, chaque utilisateur est invité à changer son mot de passe « init » par un mot de passe personnalisé.

Ainsi, le système effectue plusieurs contrôles sur le login/password avant de fournir un accès au menu des fonctionnalités du site web.

Chaque login est unique et permet d'identifier chaque utilisateur du système (en cas d'homonymes de deux utilisateurs, un login distinct sera attribué à chacun).

Code Retour	Message affiche ecran	Commentaires
1	Vous êtes connecté au système EvalNotes	Le système a authentifié et validé l'utilisateur connecté
2	Erreur : login non trouvé dans la base	Si le système ne trouve pas de login correspondant dans la base de données
3	Erreur : mauvais password saisi	S'il n'y a pas égalité parfaite entre le password saisi et le password en base
4	Première connexion : vous devez changer votre mot de passe « init »	C'est la première connexion. Il existe une autre fonction du système qui permet de changer le mot de passe « init »

Si l'authentification est OK (codeRetour égal à 1), la date de dernière connexion est mise à jour avec la date système du jour dans l'objet *Utilisateur*.

Fonctionnalités supplémentaires

Outre la gestion des notes, le système offre quelques fonctionnalités spécifiques réservées au proviseur ou aux professeurs.

Le proviseur utilise surtout le système pour avoir des statistiques et faire des actes de gestion (tutorat).

Statistiques Notes

- Le proviseur souhaite connaître la liste des élèves en grande difficulté. A ce titre, au troisième trimestre de l'année scolaire, le proviseur souhaite connaître la liste des élèves ayant des notes faibles en se basant sur les notes des deux premiers trimestres. Ainsi, le système offrira une méthode (*elevesAvecMoyenneInferieure*) donnant la liste des élèves ayant une moyenne générale inférieure à une valeur donnée pour un trimestre donné.

- Lors d'une réunion parents/professeur organisée un soir dans l'établissement, des parents ayant des jumeaux dans des classes différentes (4ème B et 4ème F) se sont plaints de la notation d'un professeur d'histoire géographie pour un de leurs enfants. En effet, la moyenne de leur premier enfant au premier trimestre sur cette matière est très inférieure à celle de leur deuxième enfant (alors que les enfants auraient un niveau équivalent dixit les parents). Des cas similaires ont également été rapportés au proviseur sur d'autres matières.

  Afin d'avoir des données précises et fiables sur le sujet, le proviseur a demandé à enrichir le système informatique d'une fonctionnalité (*comparaisonNotesProfesseur*) lui donnant un état comparatif par niveau et pour une matière de la moyenne des notes attribuées depuis le début de l'année scolaire par chaque professeur (**voir annexe 1**).

Pour des raisons de lisibilité, les notes seront données sur 20 et les moyennes seront affichées avec un seul chiffre après la virgule.

Tutorat

Le proviseur a mis en place un système de tutorat entre élèves permettant à chacun des élèves d'aider un ou plusieurs autres élèves de niveau inférieur dans sa scolarité. Ce système est non obligatoire et doit juste répondre aux trois règles ci-dessous :

- 1 élève tuteur ne peut aider qu'au maximum trois autres élèves.
- 1 élève tuteur ne peut aider un autre élève que si celui-ci se trouve dans une classe d'un niveau inférieur au sien (ex : un élève de 4ème peut être tuteur d'un élève de 5ème plus un élève de 6ème). Les élèves de 6ème ne peuvent donc pas jouer ce rôle de tuteur.
- 1 élève peut ne pas avoir de tuteur, avoir un seul tuteur ou plusieurs tuteurs.

L'annexe 2 donne une représentation graphique de cette fonctionnalité de tutorat.

Travail à faire

Question 1

La classe *AuthentificationProcessus* décrit les données et les traitements liés à l'authentification des utilisateurs sur le site web.

Ecrire en pseudo code la méthode connection qui permet d'authentifier un utilisateur sur le site (signature fournie dans la description de la classe) :

// Entier connection (p_login, p_password).

Si l'authentification a réussi, la méthode assignera l'utilisateur à l'attribut de la classe *AuthentificationProcessus*.

Question 2

Il s'agit ici d'implémenter les fonctionnalités de base du système à savoir la gestion des notes scolaires. Ces fonctionnalités sont offertes dans la classe *NotesProcessus*.

Question 2.a

Implémenter la méthode *recupererListeNotesMatiereEleve*, en pseudo-code, qui permet de restituer la liste des notes d'un élève sur une matière et pour un trimestre (à partir du numéro de trimestre).

Question 2.b

Traduire dans le langage de votre choix, l'algorithme de la question n° 2.a.

Question 3

La classe *AdminProcessus* du SI regroupe les traitements permettant de fournir des informations sur le niveau scolaire des collégiens. Elle gère aussi le mécanisme de tutorat. Ainsi, il a été développé la méthode *elevesAvecMoyenneInferieure.*

(il n'est pas demandé de donner le code de cette méthode)

Question 3.a

A partir de cette méthode, écrire en pseudo code l'instruction (donc sur 1 ligne) qui permet d'avoir la liste des élèves (sous forme de TableauObjet<Eleve>) ayant eu moins de 7/20 au premier trimestre ET au deuxième trimestre (on pourra utiliser la variable admin comme objet de la classe *AdminProcessus* et utiliser la méthode *objetsEnCommun* de la classe TableauObjet).

Question 3.b

Ecrire en pseudo code la méthode *comparaisonNotesProfesseur* qui permet au minimum de générer le corps de la page html de l'**annexe 1**.

Vous pourrez pour ce faire utiliser la classe utilitaire HashNote() et vous aider des méthodes implémentées dans la classe ChaineUtils.

Question 3.c
Ecrire la méthode *recupererEleveTuteurs*, en pseudo code, permettant la restitution des tuteurs d'un élève. Il sera demandé d'optimiser l'algorithme afin de parcourir un nombre minimum d'éléments de la liste des élèves du collège.

Par exemple, sur le schéma de l'**annexe 2**, le résultat de cette méthode donnerait :

Eleve	Tuteurs
Laurent Martie	Michel Valent
Sandrine Aubagnac	Michel Valent
Alain Landais	Michel Valent, Damien Cordet
Adrien Fort	Sylvie Testut, Damien Cordet
Murielle Landru	Sylvain Laurier
Michel Valent	
Damien Cordet	Magalie Vineau
Valentine Michaud	
Magalie Vineau	Sylvain Laurier
Eric Laurence	
Virginie Letetrel	Pierre Lauriou
Sylvain Laurier	
Anne Duverger	
Pierre Lauriou	
Sylvie Testut	

Il n'est pas demandé d'avoir un tri spécifique dans la liste des tuteurs retournés.

Question 3.d
Ecrire la méthode *ajouterEleveTuteur*, en pseudo code. Cet objet modifié (premier paramètre de la méthode) devra être sauvé en base.

Question 4

Un analyste propose de modifier légèrement le diagramme de classes des objets persistants. En effet, il remarque qu'il existe plusieurs attributs communs entre la classe Eleve et la classe Utilisateur. Il propose donc d'ajouter une relation d'héritage entre ces deux classes (Eleve hériterait de Utilisateur) et de supprimer les attributs communs dans la classe Eleve.

Trouvez-vous que c'est une bonne idée ? Argumentez.

Classes du SI

Voici quelques informations complémentaires utiles aidant à une meilleure compréhension du diagramme de classe fourni en **annexe 3**.

Les getters et les setters (méthodes get/set) sont fournies par les classes mais ne sont pas représentées dans ces diagrammes pour une meilleure lisibilité.

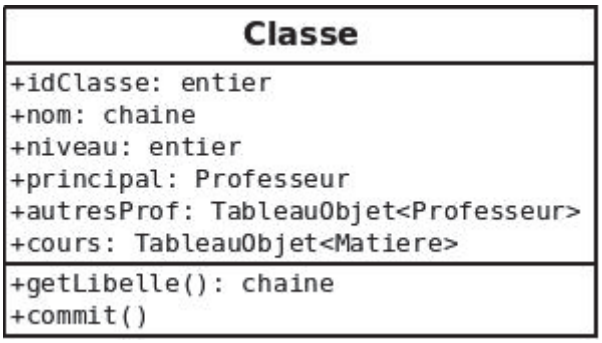
Toutes les classes disposent d'un constructeur (sans paramètre) et ne sont pas représentés dans ces diagrammes pour une meilleure lisibilité.

Classes du SI (couche persistance)

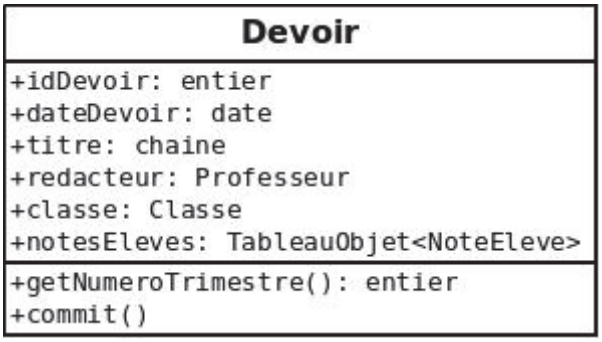
Toutes les classes ont un attribut *idxxx* qui sert d'identifiant technique permettant de s'assurer de l'unicité de chaque objet dans la base.

Ces classes ne contiennent que très peu de traitements et représentent les données du système qui sont sauvegardées dans un SGBD Relationnel.

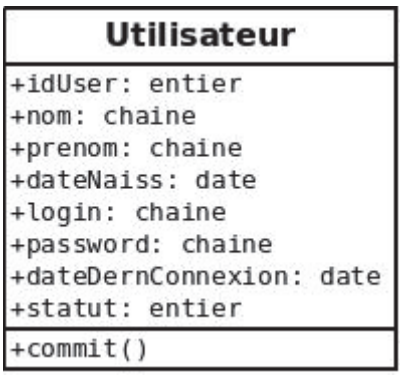
La méthode *commit*, présente dans une majorité de ces classes, permet d'enregistrer l'objet au sein du SGBDR à travers l'exécution de requêtes SQL ad hoc (insert, update).



getLibelle() : fournit un affichage complet du nom de la classe. Ex : pour nom=B et niveau=4, renverra la chaîne « 4ème B ».



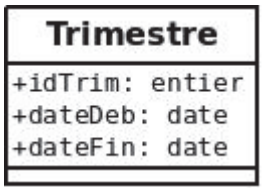
getNumeroTrimestre() : renvoie un entier 1 ou 2 ou 3 en fonction de la dateDevoir qui est incluse dans les bornes de date du trimestre.



Descriptif : toute personne qui se connecte au site web est considéré comme un utilisateur (à noter qu'un élève n'est pas un utilisateur). On considère que le couple nom/prenom est unique.

dateDernConnexion : le système enregistre automatiquement la date de dernière connexion. Il sera ainsi possible de connaître les utilisateurs (parents) qui se sont peu ou pas du tout connectés.

statut : vaut 1 pour un parent et 2 pour un prof.



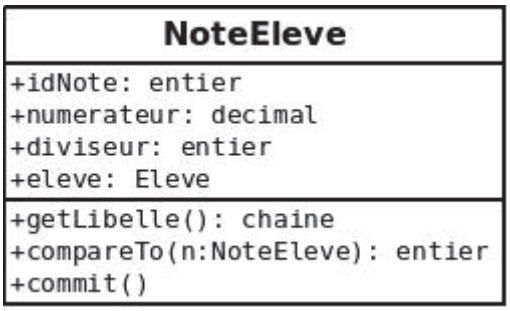
Descriptif : une année est découpée en trois trimestres scolaires permettant de répartir les devoirs sur le premier, deuxième ou troisième trimestre. Le système est initialisé avec ces trois objets qui sont immuables.

idTrim : représente le numéro du trimestre (1,2,3). Cet identifiant technique représente donc exceptionnellement une donnée fonctionnelle.

Les objets suivants sont accessibles via la méthode getTrimestre de la classe BaseDeDonnees.

Trim1 : Trimestre(1,01/09/2015,30/11/2015)
Trim2 : Trimestre(2,01/12/2015,28/02/2016)
Trim3 : Trimestre(3,01/03/2016,15/06/2016)

Aucun devoir n'est donné du 16/06 au 30/06.



Descriptif : représente une note d'un élève donné pour un devoir. Les notes, selon l'importance des devoirs, peuvent être sur 5 (ex 3/5), sur 10 (ex : 7.5/10) ou sur 20 (ex : 17.5/20).

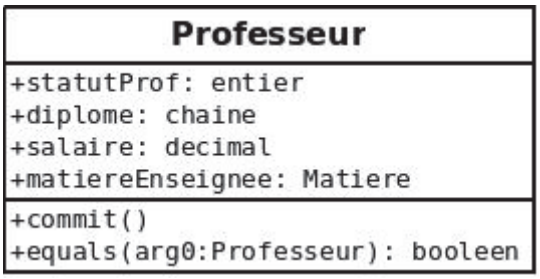
diviseur : égal à 5 ou 10 ou 20.

getLibelle() : renvoie une chaîne représentant la note. Exemple 15.5/20.

compareTo : méthode de comparaison entre 2 notes selon le quotient (rapport du numérateur sur le diviseur).

Descriptif : représente les matières enseignées aux élèves (Math, Français, Anglais, etc.).

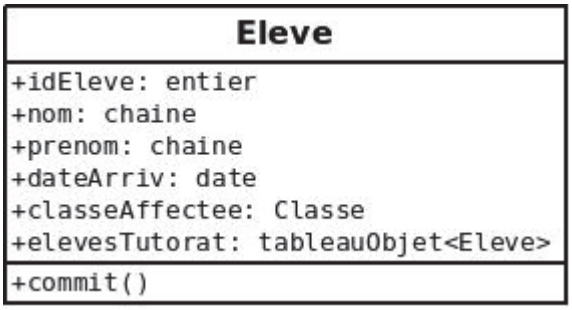
compareTo : compare 2 objets entre eux, permet d'utiliser une fonction de tri sur un TableauObjet<Matiere> dans une méthode utilitaire de tri. La comparaison se fait sur la propriété nom. Cette méthode retourne -1 si nom<mat.nom, 0 si nom == mat.nom et +1 si nom>mat.nom.



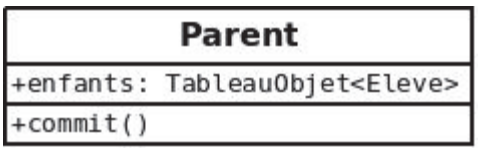
Descriptif : un professeur n'enseigne qu'une seule matière. Il peut donner des cours à plusieurs classes dans cette matière. Le proviseur est considéré comme un professeur avec un id réservé égal à 1.

statutProf : vaut 1 pour titulaire, 2 pour stagiaire, 3 pour contractuel, 4 pour proviseur.

equals : méthode utilisée par la classe HashNote lors d'une recherche dans la table associative avec une clef de type Professeur.

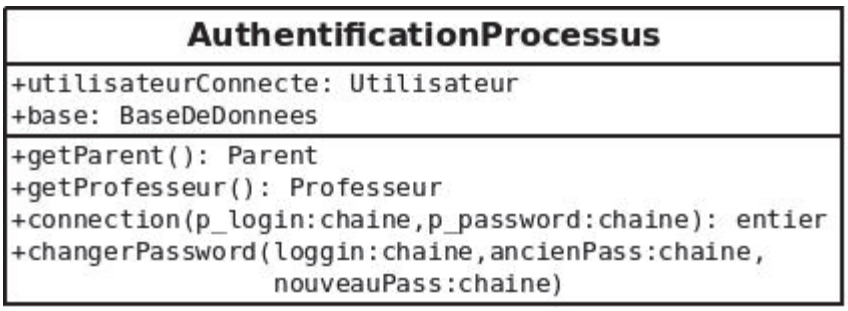


elevesTutorat : liste des élèves qui sont aidés par l'élève(tuteur).



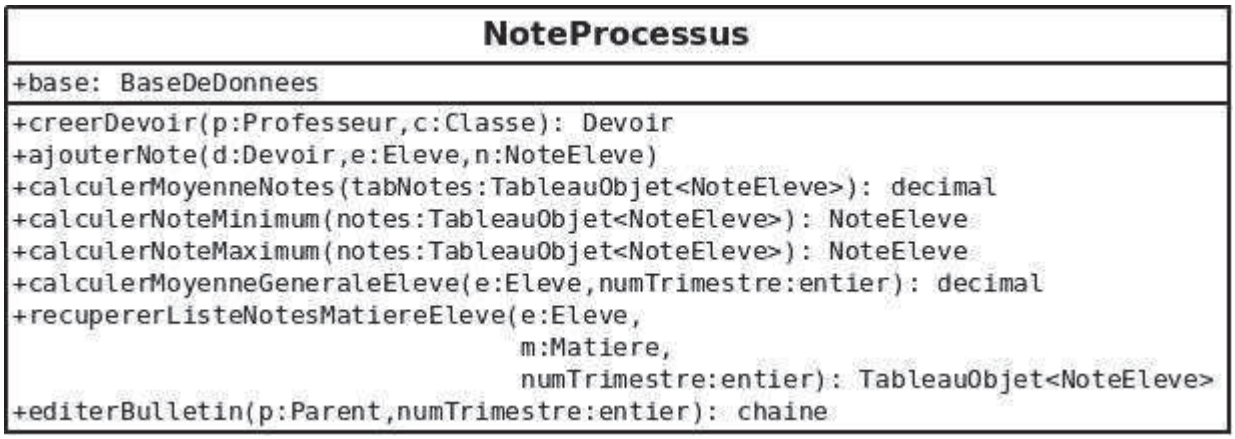
Classes du SI (couche métier)

Cette couche contient plusieurs classes processus.



connection : méthode d'authentification d'un utilisateur (question 1).

changerPassword : méthode permettant d'enregistrer un nouveau mot de passe pour un utilisateur.



base : permet au processus d'accéder aux données de la couche persistance.

creerDevoir : méthode permettant à un professeur de créer un nouveau devoir.

ajouterNote : méthode permettant d'ajouter une note à un élève pour un devoir donné.

calculerMoyenneNotes : retourne la moyenne arithmétique des notes passées en paramètre.

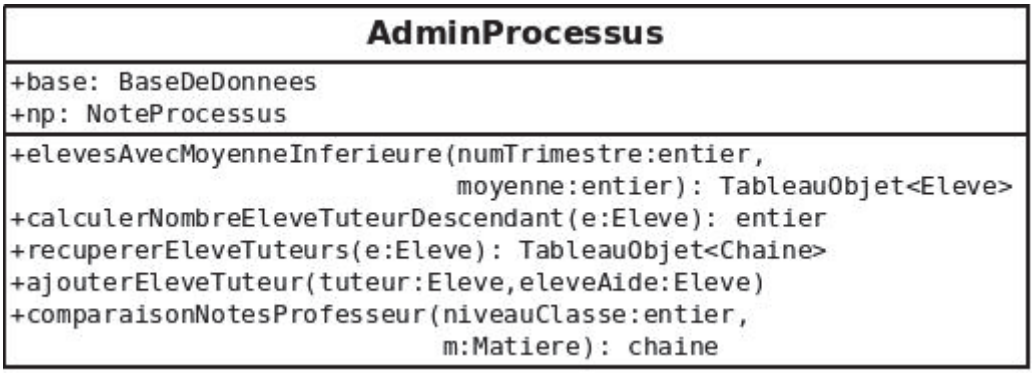
calculerNoteMinimum : retourne la note la plus faible parmi celles fournies en paramètre (exemple la note 18/20 est plus faible que 5/5).

calculerNoteMaximum : retourne la note la plus forte parmi celles fournies en paramètre (exemple la note 5/5 est plus forte que 18/20).

calculerMoyenneGeneraleEleve : retourne la moyenne arithmétique générale de l'élève. Il s'agit en fait de la moyenne de toutes les moyennes des matières enseignées dans la classe de l'élève. Toutes les matières de la classe ont le même coefficient.

recupererListeNotesMatiereEleve : retourne la liste des notes obtenues par l'élève dans une matière donnée pour un trimestre donné (question 2.a).

editerBulletin : retourne une page html représentant les bulletins trimestriels des enfants d'un parent.



elevesAvecMoyenneInferieure : retourne la liste des élèves ayant une moyenne générale sur un trimestre inférieure ou égale à la moyenne donnée en paramètre.

calculerNombreEleveTuteurDescendant : calcule le nombre d'élèves aidés directs et indirects pour un élève tuteur donné.

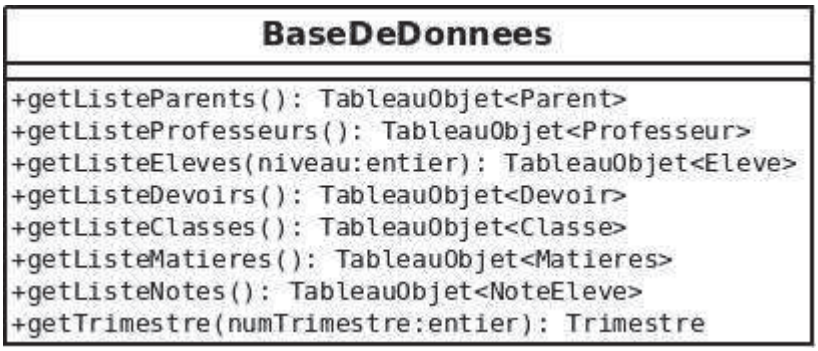
recupererEleveTuteurs : retourne la liste des tuteurs directs pour un élève donné (question 3.c).

ajouterEleveTuteur : ajoute un élève aidé à un élève tuteur (question 3.d).

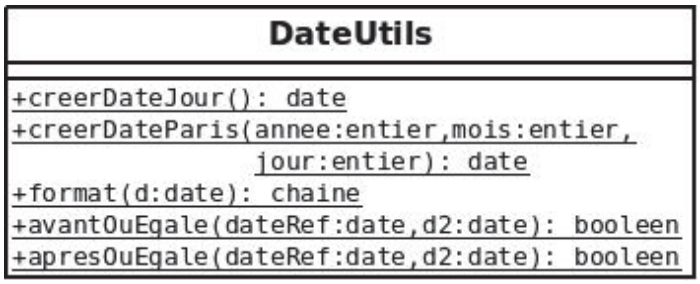
comparaisonNotesProfesseur : crée un rapport html permettant de comparer la notation des professeurs pour une matière et un niveau de classe (question 3.b).

Classes Utilitaires

Ces classes sont fournies pour aider le candidat à écrire le pseudo code des méthodes.



getListeEleves : retourne une liste des élèves par niveau de classe ($6^{ème}$, $5^{ème}$, $4^{ème}$ ou $3^{ème}$). Par exemple, getListeEleves (5) retourne tous les élèves de classe de $5^{ème}$.



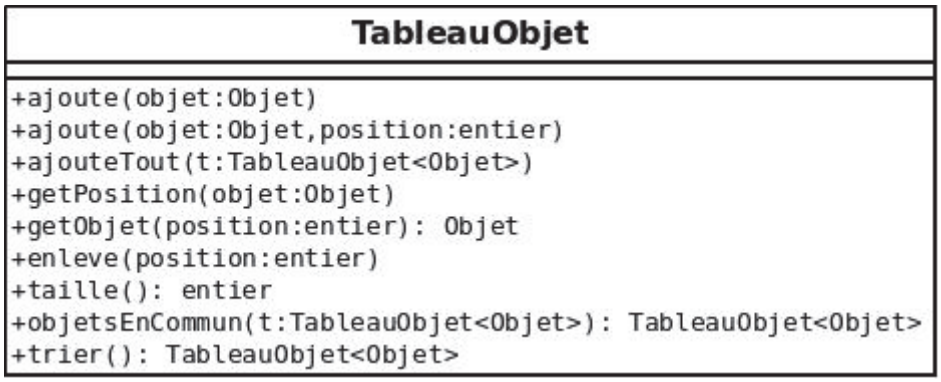
Toutes ces méthodes sont des méthodes de classe (pas d'instanciation d'objets).

format : retourne une chaîne de la forme dd/mm/yyyy de la date passée en paramètre.

avantOuEgale : retourne vrai si dateRef est antérieure ou égale à d2.

apresOuEgale : retourne vrai si dateRef est supérieure ou égale à d2.

creerDateParis : crée un objet Date avec un jour/mois/année égal aux paramètres fournis avec le fuseau horaire de Paris. Pour le mois, 0 correspond à janvier et 11 à décembre.



ajoute(objet) : ajoute un nouvel objet à la fin du tableau.

ajoute(objet, position) : ajoute l'objet à la position passée en paramètre.

ajouteTout : ajoute tous les éléments du tableau transmis en paramètre à l'objet TableauObjet.

getPosition : donne la 1ère position d'un objet dans le tableau (la 1ère position dans un tableau commence à 0) et renvoie 0 si l'objet n'y est pas.

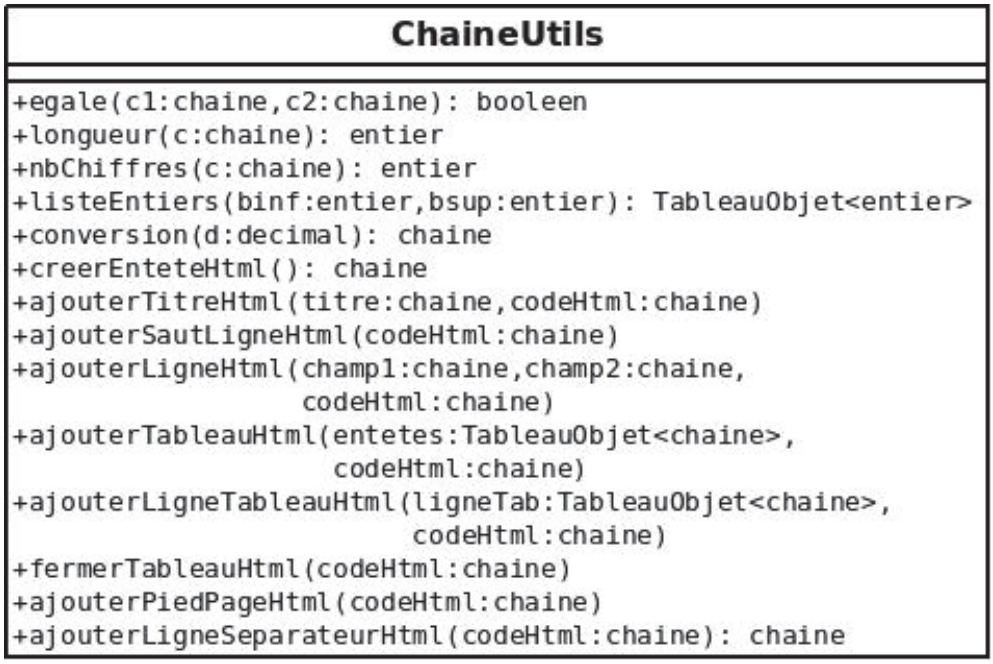
getObjet : retourne l'objet situé à la position passée en paramètre.

enleve : supprime l'objet situé à la position passée en paramètre.

taille : retourne le nombre d'éléments (objets) du TableauObjet

objetsEnCommun : retourne un nouveau tableau d'éléments (objets) tel que ces éléments font à la fois parti du TableauObjet et du TableauObjet passé en paramètres.

trier : retourne un TableauObjet trié. La méthode de tri est celle implémentée dans la méthode compare*To*(Objet o) contenue dans la classe de l'objet.



egale : renvoie vrai si les deux chaînes de caractères sont identiques.

longueur : retourne la longueur de la chaîne.

nbChiffres : retourne le nombre de chiffres contenus dans la chaîne.

listeEntiers : retourne un tableau d'objets d'entiers compris entre la borne inférieure (binf) et la borne supérieure (bsup). Si bsup<binf alors retourne null.

conversion : convertit un décimal (reel, double, float) dans une chaîne affichable avec un seul décimal affiché (ex 12.37543 retournera 12.3).

creerEnteteHtml : retourne le code html pour créer le début d'une page html.

ajouterTitreHtml : ajoute un titre à la page html.

ajouterSautLigneHtml : ajoute un saut de ligne dans la page html.

ajouterLigneHtml : ajoute deux champs sur une même ligne dans la page. Le premier champ est en gras.

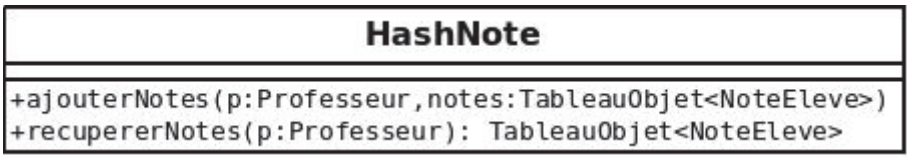
ajouterTableauHtml : ajouter un tableau html de n colonnes avec des entêtes. n est la taille du tableau passé en paramètre qui contient le nom des entêtes.

ajouterLigneTableauHtml : ajoute une ligne dans le tableau html.

fermerTableauHtml : clôt le tableau html.

ajouterLigneSeparateurHtml : ajoute une ligne horizontale dans la page html.

ajouterPiedPageHtml : clôt la page html.



Description : gère une table associative (clef/valeur) entre un professeur(clef) et un tableauObjet<NoteEleve>(valeur).

Pour rappel, une table associative est un ensemble de clef/valeur tel qu'il est possible de retrouver directement un objet valeur à partir de sa clef.

ajouterNotes : ajoute un tableauObjet<NoteEleve> à un professeur. Si la clef p n'est pas présente dans la table associative alors on crée un nouvel objet TableauObjet<NoteEleve> qu'on associe à la clef p. Si la clef p est déjà présente dans la table associative, on ajoute le tableauObjet<NoteEleve> transmis en paramètre au tableauObjet déjà associé à p.

recupererNotes : récupérer toutes les notes qui ont été ajoutées à un professeur via la méthode ajouterNotes.

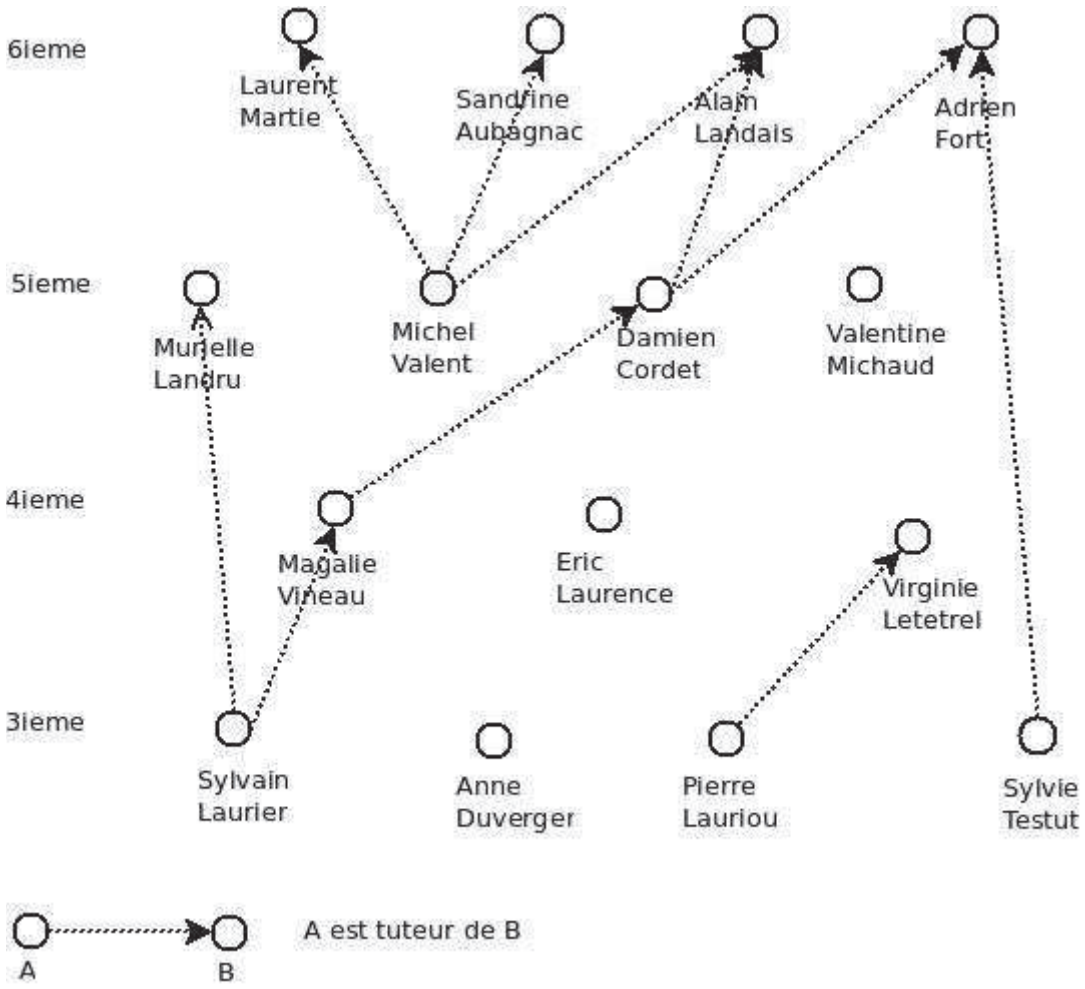
Annexe 1

Comparatif Notes Professeur

Date Etat : 12/11/2015
Niveau : 5ème
Nom Matière : Français

Professeur	Note min	Note max	Moyenne
Durant michel	5	14	10.2
Martin laurent	11	16	13.8
Cartel lysiane	11	17.5	14.5

Annexe 2


6ieme
Laurent Martie
Sandrine Aubagnac
Alain Landais
Adrien Fort
5ieme
Murielle Landru
Michel Valent
Damien Cordet
Valentine Michaud
4ieme
Magalie Vineau
Eric Laurence
Virginie Letetrel
3ieme
Sylvain Laurier
Anne Duverger
Pierre Lauriou
Sylvie Testut
A
B
A est tuteur de B

Annexe 3

Diagramme de classe des objets persistants en base de données.