



DEPARTEMENT MATHÉMATIQUE ET INFORMATIQUE
GENIE DU LOGICIEL ET SYSTEMES INFORMATIQUES DISTRIBUES

RAPPORT :

APPLICATION DE GESTION D'UNE ECOLE PRIVEE EN JAVA

ENCADRE PAR : PROF.NAJI ABDELWAHAB
REALISE PAR : MOSTAFA ASHARF



Table des matières

Pourquoi Génie Logiciel ?.....	3
Définition :	3
Les deux familles des cycles de vie	6
Le modèle choisi et pourquoi ?	8
Les différentes phases.....	10
Spécification des besoins	10
Conception	10
Implémentation	10
Testes.....	11
Analyse incrément 1 :	13
Conception incrément 1 :	15
Implémentation incrément 1 :	19
Remarque incrément 1:.....	20
Analyse incrément 2 :	23
Conception incrément 2 :	24
Implémentation incrément 2 :	28
Remarque incrément 2 :	31
Analyse incrément 3 :	33
Implémentation incrément 3 :	35



Pourquoi Génie Logiciel ?

Commençant d'abord par la définition de Génie Logiciel. C'est quoi Génie Logiciel ?

Définition :

Génie Logiciel, ou bien 'software engineering', est l'ensemble des méthodes et des techniques employées par une équipe ou une entreprise spécialisée au développement des logiciels. Pour garantir la conception, le développement et la maintenance des systèmes informatiques de bonne qualité en consommant la moindre des ressources possibles (temps, coût de maintenance... etc.).

Le génie logiciel touche au cycle de vie des logiciels, l'analyse du besoin, l'élaboration des spécifications, la conceptualisation du mécanisme interne au logiciel ainsi que les techniques de programmation, le développement, la phase de test et finalement la maintenance.

Le génie logiciel est « l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi ». [Journal officiel du 19 février 1984]



La crise du logiciel en 1968 :

Le terme software engineering a été mentionnée pour la première fois en 1967 et repris l'année suivante à une conférence concernant la crise du logiciel.

La crise du logiciel est une baisse significative de la qualité des logiciels dont la venue coïncide avec le début de l'utilisation des circuits intégrés dans les ordinateurs. Cette augmentation de puissance des compétences de calcul des ordinateurs a permis de réaliser des logiciels beaucoup plus complexes qu'auparavant.

Mais les premières tentatives de création de logiciels de grande ampleur ont vite montré les limites d'un travail informel d'ingénieurs logiciel.

Ce qui conduit à des produits qui ne sont pas terminés dans le temps, et coûtent plus cher en entretien. La baisse du coût du matériel informatique s'accompagnait d'une augmentation du coût du logiciel. Des études se sont penchées sur la recherche de méthodes de travail adaptées à la complexité inhérente aux logiciels contemporains et ont donné naissance au génie logiciel dans les années 1970 sous la coordination de l'OTAN.



Pourquoi Génie Logiciel :

Chaque année plusieurs milliards de dollars sont perdus dans l'industrie du logiciel. D'ailleurs les chiffres révélés par une étude menée par le Standish Group sur le comportement des projets informatiques le prouvent. L'étude menée en 1995 sur 8380 projets révèle que :

- 16% des applications sont construites avec succès ;
- 53% des projets aboutissent mais posent des problèmes tels : la diminution des fonctionnalités fixées de départ, le non-respect des délais spécifiés ou encore l'augmentation des coûts ;
- 31% des projets sont purement et simplement abandonnés.

Le Génie Logiciel est venu pour : garantir la qualité du produit et le respect des besoins de l'utilisateur tout en respectant les délais et les coûts fixés au départ.



Les deux familles des cycles de vie

Il existe deux modèles de cycle de vie :

- 1- Les modèles classiques
- 2- Les modèles agiles

C'est quoi un cycle de vie d'un logiciel ?

Le processus de production des logiciels passe par différentes étapes enchainées, en appel l'ensemble de ces étapes un cycle de vie.

Le cycle de vie représente un plan de production, une forme de guide pour les développeurs logiciels. La construction d'un logiciel ne doit pas s'effectuer d'une manière aléatoire, si on veut garantir la production d'un logiciel qui répond aux besoins des clients il va falloir qu'on suit une structure un processus définie par un ensemble d'étapes bien précis.

Les principales étapes dans un cycle de vie sont les suivants :





Il existe différents modèles de cycle de vie, en les regroupe généralement en deux grandes familles :

Les modèles classiques :

- Modèle en cascade (Waterfall Model)
- Modèle itérative (Iterative Model)
- Modèle en spirale (Spiral Model)
- Modèle en V (V-Model)
- Big Bang Model

Les modèles classiques se sont des modèles stricts, où les étapes sont bien claires et spécifiques, l'utilisation des modèles classiques nécessite une documentation bien détaillée.

Les modèles classiques fonctionnent convenablement pour les grands projets où les besoins sont bien définis et stables.



Les modèles agiles :

Le modèle choisi et pourquoi ?

Le choix du modèle de cycle de vie se fait selon les critères suivants :

- Selon la nature et la taille du projet
- Selon les compétences et l'effectif de l'équipe
- Selon les besoins du client

L'idée du projet consiste à réaliser une application de gestion d'une école privée en JAVA.

Selon la nature du projet et le type du client, j'aurais besoins d'une méthode de construction qui se caractérise par la fiabilité, la rapidité, et l'adaptabilité aux besoins des clients.

Je n'ai pas eu le temps pour faire la documentation, au plus je n'ai pas assez d'expérience en programmation java, alors le modèle utilisé ne doit pas nécessiter une maîtrise de java.

En analysant les caractéristiques de ce projet on trouvera que mes clients se sont les écoles privées, la taille du projet est grande, or que la présence du système est essentielle dans l'école, par exemple : on ne pourra pas faire inscrire un nouvel élève sans que le module inscription soit prêt et validé par le client.



Donc se sera judicieux si j'aurais pu construire mon système sous forme de module de tel sort que chaque partie doit être fonctionnelle et testable et utilisable par mon client, puisque le système est assez grand, le client va attendre beaucoup de temps pour avoir la totalité de son système. En appliquant cette méthode de découpage '**diviser pour régner**' je vais satisfaire mon client en lui offrir à chaque fois une partie fonctionnelle.

Devant toutes ces contraintes et selon les trois critères que j'ai cité ci-dessus j'ai choisi de travailler avec le modèle '**Incrémental**', par ce qu'il permettra le développement rapide d'une partie du système, les besoins de mes clients sont claires et stables, donc le modèle incrémental mérite d'être utilisé dans ce projet.

Le module d'inscription sera le premier à construire par ce qu'il pose beaucoup de problème, de plus c'est un élément essentiel de mon système.



Les différentes phases

Au premier temps je dois décomposer le système en plusieurs modules, et pour chaque module je vais faire les phases suivantes :

Spécification des besoins

Déterminer les besoins du client pour le module X.

Conception

Dans cette phase, je vais préciser les différents diagrammes :

- Diagramme de cas d'utilisation
- Diagramme de classe métier
- Diagramme de classe de conception
- Diagramme de séquence système
- Diagramme de séquence d'interaction

L'utilisation de ces diagrammes est indispensable par ce qu'ils représentent un résumé universel des besoins exprimés par le client c'est un support de communication commun entre tous les développeurs afin de faciliter la communication le travail d'équipe.

Implémentation

Cette phase représente généralement le codage



Testes

La phase des tests est très importante dans le processus de développement des logiciels, afin de fournir au client un logiciel de qualité, chaque module de mon système sera tester à part.

Ilya deux types de teste :

- **Teste Unitaires** : ce sont les tests effectués au cours du cycle de vie de chaque module.
- **Teste de d'intégration** : avoir terminer les différents modules il fallut tester l'ensemble de ces modules, c'est-à-dire tester l'intégralité de mon système.

On pourra schématiser le modèle '**Incrémental**' par l'image suivante :

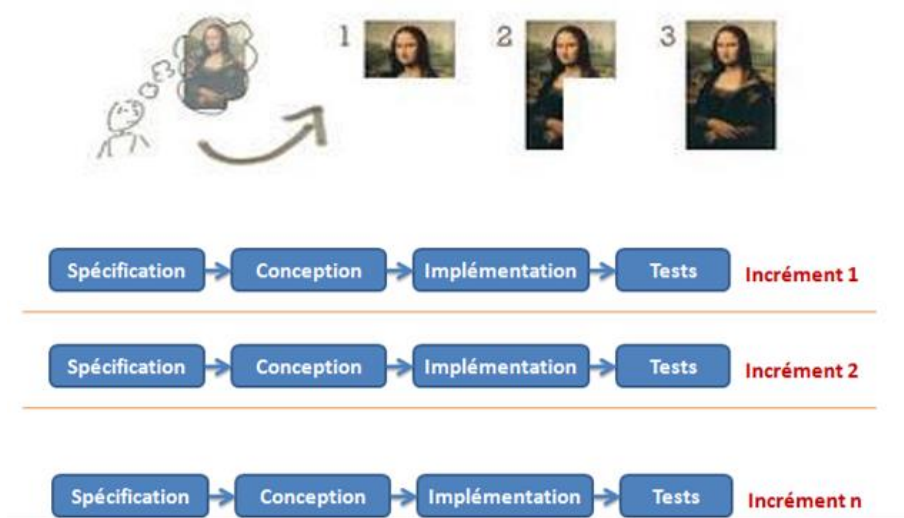


Figure 1 : cycle de vie incrémentale



INCREMENT 1



Analyse incrément 1 :

Mon client a besoins d'un module qui prend en charge l'inscription et l'affectation des élèves aux groupes. Les élèves peuvent être inscrivent dans un ou plusieurs niveaux, et chaque niveau peut contenir un ou plusieurs groupes. Le nombre des élèves dans un groupe ne peut pas dépasser 32 élèves. L'utilisation de ce module sera effectuée par un agent de saisie. Chaque année universitaire, l'élève est inscrit dans un niveau.

L'agent de saisie prend en charge l'inscription des élèves, il peut ajouter et supprimer une inscription. Il peut rechercher un élève par ses informations personnels (nom, prénom, ... etc.) ou par niveau, et groupe.

Un élève est caractérisé par son code d'étudiant (cne), nom, prénom et adresse.

L'agent de saisie possède un compte par lequel il peut gérer l'inscription

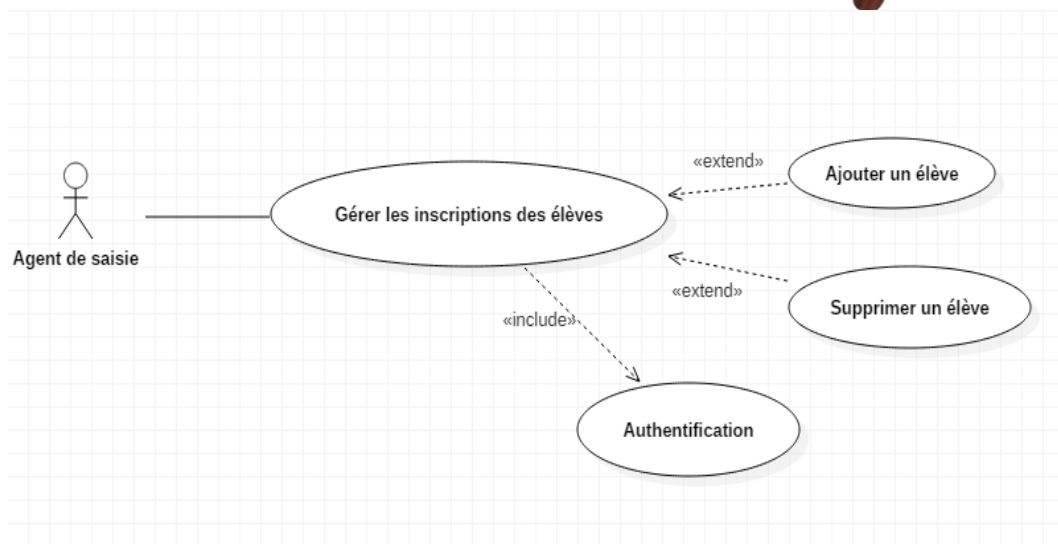


Figure 2 : use case incrément 1

Ici comme vous voyez dans le diagramme de cas d'utilisation, l'agent de saisie peut gérer les inscriptions c'est-à-dire ajouter ou supprimer une inscription, ces fonctionnalités nécessitent l'authentification de l'agent de saisie.



Conception incrément 1 :

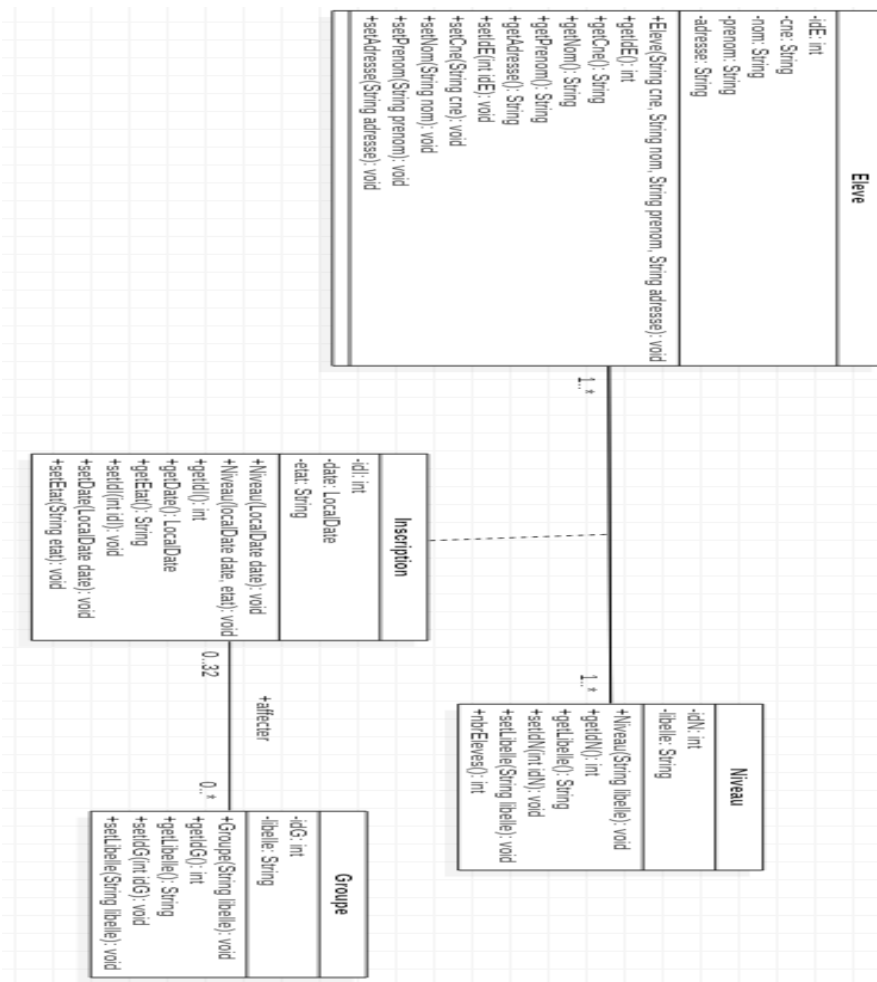


Figure 3 : diagramme de classe métier incrément 1



Un élève peut faire plusieurs inscriptions dans plusieurs niveaux, chaque inscription est affectée à un seul groupe.

Un élève est caractérisé par : cne, nom, prénom et adresse.

Chaque niveau à un libelle, le groupe aussi.

L'inscription est définie par une date d'inscription et l'état d'inscription.



Figure 4 : diagramme de classe conception incrément 1



Le diagramme de classe de conception ne présente pas que la couche métier, il présente la conception de l'application. Via l'utilisation du DAO, chaque classe à sa propre classe dao, cette classe dao, par exemple daoEleve prend en charge l'interaction avec la base de données, via l'utilisation de DAO on peut appliquer la philosophie MVC, Modèle View Controller. Toutes les classes dao héritent de la classe mère DAO, qui de son tour implémente l'interface IDAO, où les méthodes de gestion son définis.

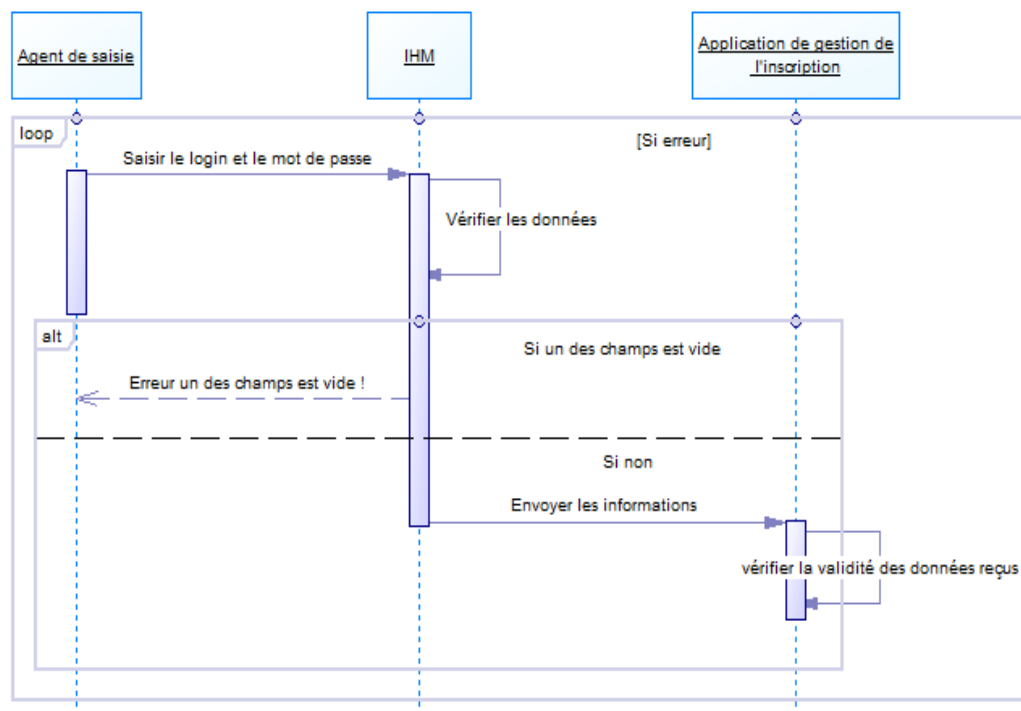


Figure 5 : diagramme de séquence système incrément 1



Implémentation incrément 1 :

```
public void delete(ActionEvent event) throws IOException {

    if (mainTable.getSelectionModel().getSelectedItem() == null) {
        System.out.println("erreur");
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Erreur ! Veuillez sélectionner une ligne");
        alert.setHeaderText(null);
        alert.setContentText("On ne peut pas supprimer le vide !!");
        alert.showAndWait();
    } else {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Warning ! Action de suppression");
        alert.setHeaderText(null);
        alert.setContentText("Voulez-vous vraiment supprimer cette inscription ?");
        alert.showAndWait();

        if (alert.getResult() == ButtonType.OK) {

            int i = index.get();

            if (i > -1) {
                DaoEleve daoEleve = new DaoEleve();
                DaoInscription daoInscription = new DaoInscription();
                boolean s1 =
daoEleve.supprimer(listeAllInscriptionsDetails.get(i).getIdE());
                boolean s2 =
daoInscription.supprimer(listeAllInscriptionsDetails.get(i).getIdI());

                if (s1 && s2) {
                    labelMessage.setText("L'inscription numéro : " +
listeAllInscriptionsDetails.get(i).getIdI() + " à bien été supprimée.");
                }
                listeAllInscriptionsDetails.remove(i);
                mainTable.getSelectionModel().clearSelection();
            }
            //vider la barre de recherche :
            searchBar.clear();
        }
    }
}
```

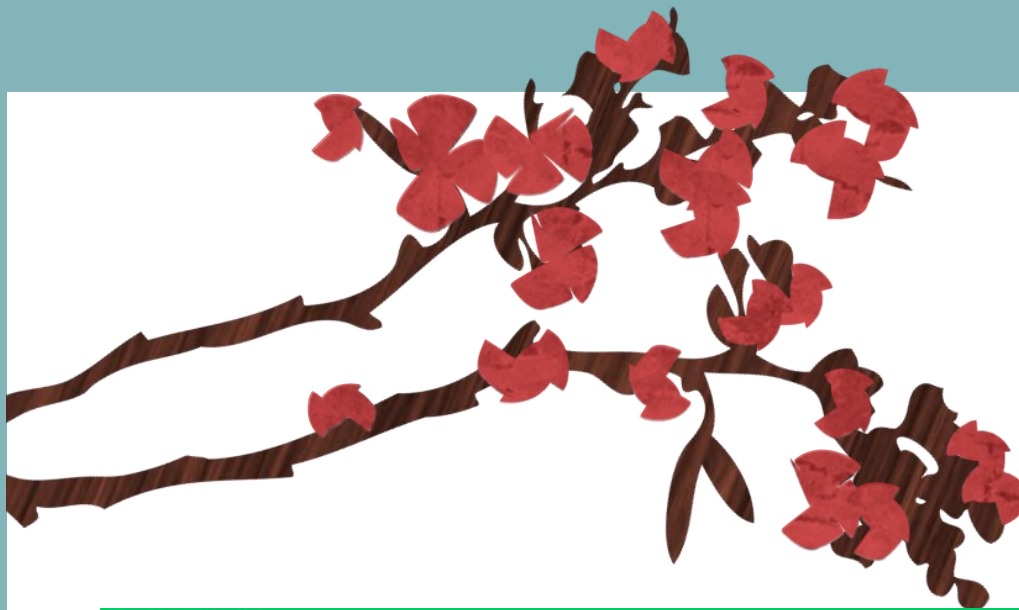


Remarque incrément 1:

La méthode delete() prend en charge la suppression d'un élève (x) sélectionné.

A screenshot of a web application window titled "Authentification". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The background of the window is a close-up photograph of a green leaf with water droplets. Overlaid on this background are two white input fields: the top one is labeled "User name" and the bottom one is labeled "Password". Below these fields are two buttons: a teal button on the left labeled "Se connecter" and a pink button on the right labeled "Annuler".

Figure 6: page d'authentification



Bienvenu sur l'accueil

Mon profile Gestion des notes Statistiques Aide

Agent de saisie > Accueil

Chercher un élève ou une inscription par ce que vous voulez

Modifier **Supprimer**

Cne	Nom	Prénom	Adresse	Date d'inscription	Etat d'inscription	Niveau	Groupe
789654	achraf	ali	maroc	2016-12-20	affectee	Niveau 2	groupe 2
1210470314	etudiant	etudiant	etudiant	2016-12-23	affectee	Niveau 3	groupe 112
123047	asdcvx	asdcvx	asdcvx	2016-12-23	affectee	Niveau 1	groupe 11
4125	swq	swq	swq	2016-11-08	affectee	Niveau 1	groupe 11
1253	pppp	pppp	pppp	2016-12-06	affectee	Niveau 1	groupe 11
1212321	teste	teste	teste	2016-12-22	affectee	Niveau 1	groupe 11
78589	dffff	dffff	dffff	2016-12-22	non affectee	Niveau 1	
152	sssss	sssss	sssss	2016-12-23	non affectee	Niveau 1	
45668665	ererere	ererere	ererere	2016-12-23	non affectee	Niveau 1	
123456	oooooooo	oooooooo	oooooooo	2016-12-22	affectee	Niveau 1	groupe 11
152647	azzzz	azzzz	azzzz	2016-12-23	affectee	Niveau 2	groupe 1
152674	asdf	asdf	asdf	2016-12-24	affectee	Niveau 2	groupe 2
1235465	mostafa	mostafa	mostafa	2016-12-23	affectee	Niveau 2	groupe 1
785	aze	aze	aze	2016-12-21	affectee	Niveau 2	groupe 2
1526354855	Rahaf	ASHARF	BOUZNIKA	2016-12-17	affectee	Niveau 3	groupe 112
1210370315	Mostafa	ASHARF	BOUZNIKA	2016-12-17	affectee	Niveau 3	groupe 11
74196380	john	john	john	2016-12-17	non affectee	Niveau 3	

CNE
NOM
PRENOM
ADRESSE
12/27/2016
Niveau 1
Ajouter **Vider**

Status : Aucune opération pour le moment...

Figure 7: page d'accueil



INCREMENT 2



Analyse incrément 2 :

L'agent de saisie peut ajouter, modifier ou consulter les notes des élèves, ou il peut imprimer la relevée des notes d'un élève sélectionné.

Chaque niveau est composé de plusieurs modules et de deux semestres, chaque module est composé de plusieurs matières, toutes ces modules est appartient à un seul semestre.

L'étudiant valide le module si toutes ses matières sont validées.

A partir des résultats des modules de semestre, le résultat de semestre est calculé. Le semestre est validé, seulement si tous ces modules sont validés.

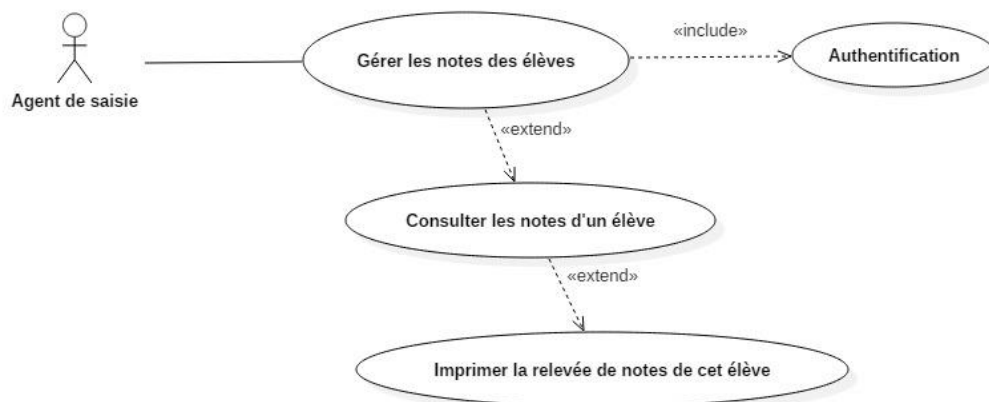


Figure 8: use case incrément 2

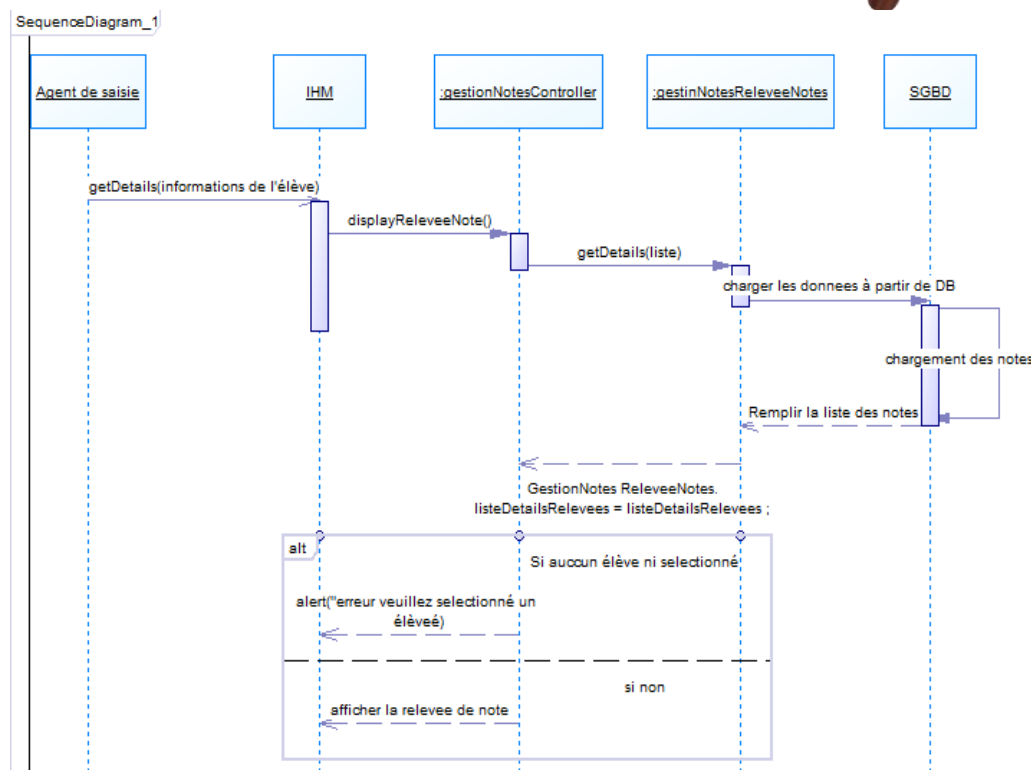


Figure 10 : diagramme de séquence système pour le cas d'affichage de la relevée de notes pour un élève (x)

Ce diagramme présente les objets qui participent dans le processus d'affichage de la relevée de notes d'un élève (X), lorsque l'agent de saisie sélectionne un élève et clique sur (détails), le controller "GestionNotesController" fais appel à la fonction "getDetails()" cette



Fonction appartient à la classe "GestionNotesReleveeNotes" cette classe fais appel au DAO c'est-à-dire le chargement des informations à partir de la base de données.

Mon profile Gestion des notes Aide Statistiques

Agent de saisie > Gestion des notes

Niveaux	Groupes	Semestre	Modules	Matières	CNE	NOM	PRENOM	NOTE	VALIDER
Niveau 1	groupe 11	S1	Développement ...	Java	7777777	sosito	sosito	19.5	V
Niveau 2	groupe 112	S2	Réseaux Informat...		8888888	bbbbbb	bbbbbb		
Niveau 3			Programmation J...		9999999	ccccccc	ccccccc		
			UML		4444444	dddddd	dddddd		
			Génie Logiciel		5555555	eeeeeeee	eeeeeeee		
			Stage de fin d'ét...		6666666	fffffffff	fffffffff		
			Projet de fin d'ét...		1111111	ggggggg	ggggggg		
					3333333	hhhhhhh	hhhhhhh		
					789789	iiiiiii	iiiiiii		
					9459549	jjjjjjj	jjjjjjj		
					963963	kkkkkkk	kkkkkkk		
					1471471	lllllllll	lllllllll		
					2582582	mmmmmmm	mmmmmmm		
					369369	nnnnnnn	nnnnnnn		
					151515	oooooooo	oooooooo		
					1210370315	Mostafa	ASHARF		

Enregistrer

Détails

Vider

Status : message

Figure 11: gestion des notes


L'agent de saisie peut consulter à tout moment la relevée de notes d'un élève donné.



Cne : 7777777

Nom : sosito

Prénom : sosito



Niveau : Niveau 3

Groupe : groupe 11

Semester :

Relevée de notes

2016/2017

Semesters	Modules	Matières	Notes	Moyennes
Semester 1	C/C++	C/C++	19.5	16.61
Semester 1	Télécommunication	Télécommunication	15.25	16.61
Semester 1	Algorithmique	Algorithmique	19.0	16.61
Semester 1	Analyse(3)	Analyse(3)	17.0	16.61
Semester 1	Probabilité	Statistique et pro...	14.0	16.61
Semester 1	Projet personnel(2)	Projet Personnel	15.0	16.61
Semester 1	Communication	Communication	16.5	16.61
Semester 2	Projet de fin d'étude	Projet de fin d'Etu...	16.75	16.36
Semester 2	UML	Modélisation UML	16.0	16.36
Semester 2	Stage de fin d'étude	Stage de fin D'étu...	17.25	16.36
Semester 2	Réseaux Informatique	Réseaux informati...	13.5	16.36
Semester 2	Programmation Java	Java	19.5	16.36
Semester 2	Génie Logiciel	Génie Logiciel	14.75	16.36
Semester 2	Développement web	Framework & CMS	17.0	16.36
Semester 2	Développement web	XML	16.5	16.36

Moyenne générale : 16.49

Imprimer

Figure 12 : exemple d'une relevée de notes

27



Implémentation incrément 2 :

```
public static void getDetails (ObservableList<GestionNotes_Semester>
listeAllSemesters, ObservableList<GestionNotes_ElevesNotes>
listeAllElevesNotes, ObservableList<GestionNotes_notes>
listeNotesFoOneEleve, SimpleIntegerProperty index_eleveNote,
SimpleIntegerProperty index_niveaux, SimpleIntegerProperty index_groupes,
SimpleIntegerProperty index_semestre,
ObservableList<GestionNotes_Niveaux> listeAllNiveaux,
ObservableList<GestionNotes_Groupes> listeAllGroupes) {

    Map<String, HashMap<String, Double>> mapNotesModulesS1 = new
HashMap<String, HashMap<String, Double>>();
    Map<String, HashMap<String, Double>> mapNotesModulesS2 = new
HashMap<String, HashMap<String, Double>>();
    ObservableList<GestionNotes_ReleveeNotes> listeDetailsRelevees_ =
FXCollections.observableArrayList();
    List<Matiere> listeMatieresOfModule = new ArrayList<Matiere>();
    List<Module> listeModulesOfSemester = new ArrayList<Module>();
    //List<GestionNotes_notes> listeNotes = new ArrayList<>();
    int i = index_eleveNote.get();
    //int niveau = listeAllNiveaux.get(index_niveaux.get()).getIdN();
    int module;
    int semestre;
    double moyenneModule = 0;
    double moyenneSemester1 = 0;
    double moyenneSemester2 = 0;
    double moyenneGenerale = 0;
    boolean moduleComplet = true;
    boolean semestreComplet = true;
    List<GestionNotes_Modules> listeNotesModules = new
ArrayList<GestionNotes_Modules>();

    //pour chaque module de S1 et S2 de l'année universitaire courante :
    for (int t = 0; t < listeAllSemesters.size(); t++) {
        semestre = listeAllSemesters.get(t).getIdS();
        listeModulesOfSemester.clear();
        listeNotesModules.clear();

        //Récupérer les modules du semestre courant :
        new DaoModule().getBySemester(semestre, listeModulesOfSemester);

        //On calcule la moyenne pour chaque module
        for (int j = 0; j < listeModulesOfSemester.size(); j++) {
            moyenneModule = 0;
```



```
module = listeModulesOfSemester.get(j).getIdM();
//mapNotesMatieres.clear();

//Récupérer les matières du module courant :
listeNotesFoOneEleve.clear();
listeMatièresOfModule.clear();
new
DaoMatiere().getByIdModule(listeModulesOfSemester.get(j).getIdM(),
listeMatièresOfModule);

//Remplir la liste 'ListeNotesFoOneEleve' qui contient les
notes de chaque matières appartienne au module courant de l'élève
choisi :

GestionNotes_ElevesNotes.getNotesEleve(listeAllElevesNotes.get(i).getCne(
), listeMatièresOfModule, listeNotesFoOneEleve, module,
LocalDate.now().getYear());

//Remplir Map par les notes des matières de S1
if (listeAllSemesters.get(t).getSemester().equals("S1")) {
    Map<String, Double> mapNotesMatieres = new
HashMap<String, Double>();
    for (GestionNotes_notes aListeNotesFoOneEleve :
listeNotesFoOneEleve)

mapNotesMatieres.put(aListeNotesFoOneEleve.getMatiere(),
aListeNotesFoOneEleve.getNote());

mapNotesModulesS1.put(listeModulesOfSemester.get(j).getLibelleM(),
(HashMap<String, Double>) mapNotesMatieres);
}

//Créer Map du deuxième semestre :
else {
    Map<String, Double> mapNotesMatieres = new
HashMap<String, Double>();
    for (GestionNotes_notes aListeNotesFoOneEleve :
listeNotesFoOneEleve)

mapNotesMatieres.put(aListeNotesFoOneEleve.getMatiere(),
aListeNotesFoOneEleve.getNote());

mapNotesModulesS2.put(listeModulesOfSemester.get(j).getLibelleM(),
(HashMap<String, Double>) mapNotesMatieres);
```



```
    }

    //Calculer la moyenne du module courant :
    for (GestionNotes_notes aListeNotesFoOneEleve :
listeNotesFoOneEleve) {
        moyenneModule += aListeNotesFoOneEleve.getNote();
    }
    if (listeNotesFoOneEleve.size() != 0)
        moyenneModule = moyenneModule /
listeNotesFoOneEleve.size();
    else
        moduleComplet = false; //si une note est introuvable, on
mentionne qu'il reste une matière sans note !

    //Remplir la liste qui rassemble la moyenne de chaque
module :
    listeNotesModules.add(new
GestionNotes_Modules(listeModulesOfSemester.get(j).getLibelleM(),
moyenneModule));
}

//Calculer la moyenne du premier semestre :
if (listeAllSemesters.get(t).getSemester().equals("S1")) {

    //Parcourir les notes des modules de S1 :
    for (GestionNotes_Modules listeNotesModule :
listeNotesModules) {
        moyenneSemester1 += listeNotesModule.getMoyenne();
    }
    moyenneSemester1 = (double) Math.round((moyenneSemester1 /
listeNotesModules.size()) * 100) / 100;
    if (!moduleComplet)
        System.out.println("module incomplet");
}

//Calculer la moyenne du deuxième semestre :
else {
    //Parcourir les notes des modules de S1 :
    for (GestionNotes_Modules listeNotesModule :
listeNotesModules) {
        moyenneSemester2 += listeNotesModule.getMoyenne();
    }
    moyenneSemester2 = (double) Math.round((moyenneSemester2 /
listeNotesModules.size()) * 100) / 100;
    if (!moduleComplet)
        System.out.println("module incomplet");
}
```




```
    }  
}  
  
    //Calculer la moyenne générale de l'élève dans l'année universitaire  
    courante :  
    moyenneGenerale = (double) Math.round(((moyenneSemester1 +  
moyenneSemester2) / 2) * 100) / 100;  
  
    String cne = listeAllElevesNotes.get(index_eleveNote.get()).getCne();  
    String nom =  
listeAllElevesNotes.get(index_eleveNote.get()).getNomEleve();  
    String prenom =  
listeAllElevesNotes.get(index_eleveNote.get()).getNomEleve();  
    String niveau =  
listeAllNiveaux.get(index_niveaux.get()).getLibelleN();  
    String groupe =  
listeAllGroupes.get(index_groupes.get()).getLibelleG();  
    String semester_ =  
listeAllSemesters.get(index_semestre.get()).getSemester();  
    String anneeUniv = Integer.toString(LocalDate.now().getYear()) + "/"  
+ Integer.toString(LocalDate.now().getYear() + 1);  
  
    //Remplir la liste des informations à afficher :  
    listeDetailsRelevees_.add(new GestionNotes_ReleveeNotes(cne, nom,  
prenom, niveau, groupe, mapNotesModulesS1, mapNotesModulesS2,  
moyenneSemester1, moyenneSemester2, moyenneGenerale));  
    GestionNotes_ReleveeNotes.listeDetailsRelevees =  
listeDetailsRelevees_;  
}
```

Remarque incrément 2 :

Ce code permet de d'avoir les notes de toutes les matières constituant une un module donné pour un élève (x), cette fonction calcule aussi la moyenne e chaque module, et chaque semestre et enfin la moyenne générale.



INCREMENT 3



Analyse incrément 3 :

En plus de ses salles de cours, l'école dispose de deux autres salles pour les séminaires. Ces deux salles peuvent être louées à la journée par des professionnels de la formation. Chaque salle peut avoir l'emploi de son occupation. La réservation d'une salle peut être faite en ligne pour une date donnée. Un client ne peut effectuer une réservation que si la salle est libre.

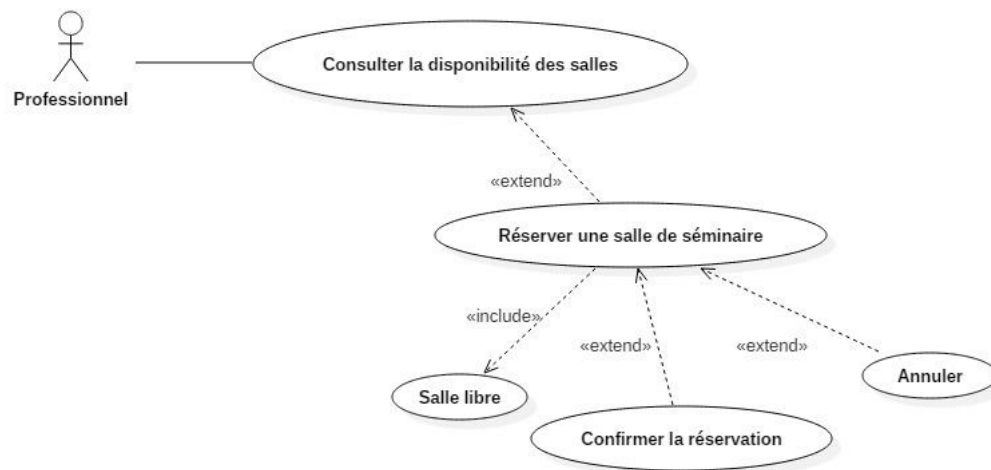


Figure 13 : diagramme de cas d'utilisation incrément 3



Dans le diagramme de cas d'utilisation on voit que les professionnels (clients) peuvent consulter la disponibilité des salles de séminaires. Au plus de la consultation les clients peuvent réserver les salles de séminaires.

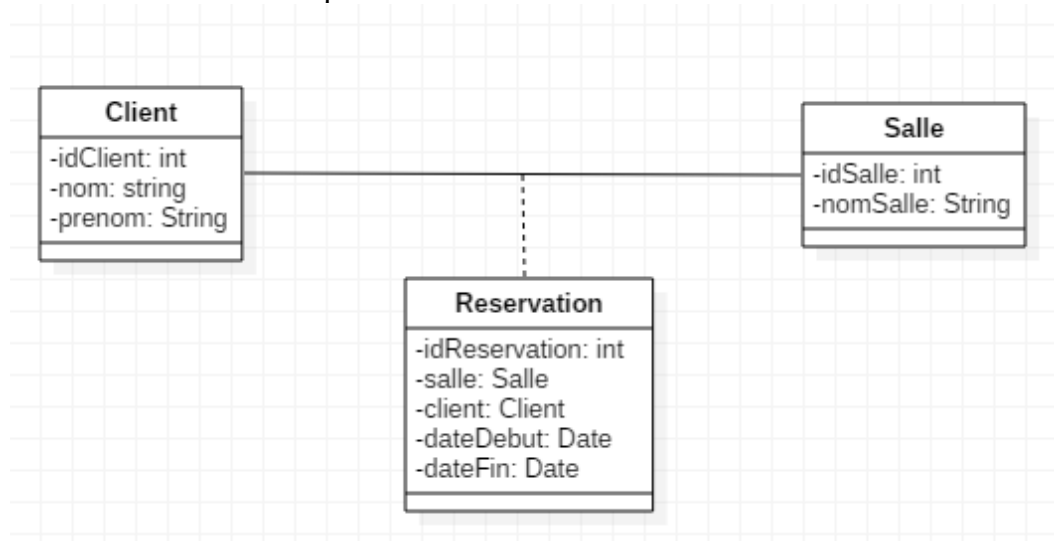


Figure 14 : diagramme de classe métier incrément 3

Les clients (généralement des professionnels du domaine d'enseignement) peuvent réserver une salle pendant une période donnée. La salle est caractérisée par un identifiant et libelle. Le client est défini par son nom, prénom, et un identifiant. Le client peut effectuer des réservations d'une salle (X), cette réservation à une date de début et date de fin.



Implémentation incrément 3 :

```
static public synchronized void reserver(Reservation reservation) {
    System.out.println(Thread.currentThread().getName() + " en cour de
reservation");

    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    DaoReservation daoReservation = new DaoReservation();
    ArrayList<Reservation> LR;
    LR = GetAllReservations(reservation.getSalle());
    boolean disponible = true;
    for (Reservation rsB : LR) {
        disponible = reservation.verifierDisponible(rsB.getDateDebut(),
rsB.getDateFin(), reservation.getDateDebut(), reservation.getDateFin());
        if (!disponible) {
            break;
        }
    }
    if (disponible) {
        daoReservation.create(reservation);
        System.out.println("Votre demande de réservation à bien été
acceptée");
    } else
        System.out.println("Désolé ! Cette salle n'est pas disponible
pour le moment");
    }
```



```

Run Program
"C:\Program Files\Java\jdk1.8.0_111\bin\java" ...
Thread-0 Demande de réservation...
Thread-1 Demande de réservation...
Thread-0 en cour de reservation
Désolé ! Cette salle n'est pas disponible pour le moment
Thread-1 en cour de reservation
Désolé ! Cette salle n'est pas disponible pour le moment

Process finished with exit code 0

```

Figure 15 : résultat de l'exécution

Diagramme de composants :

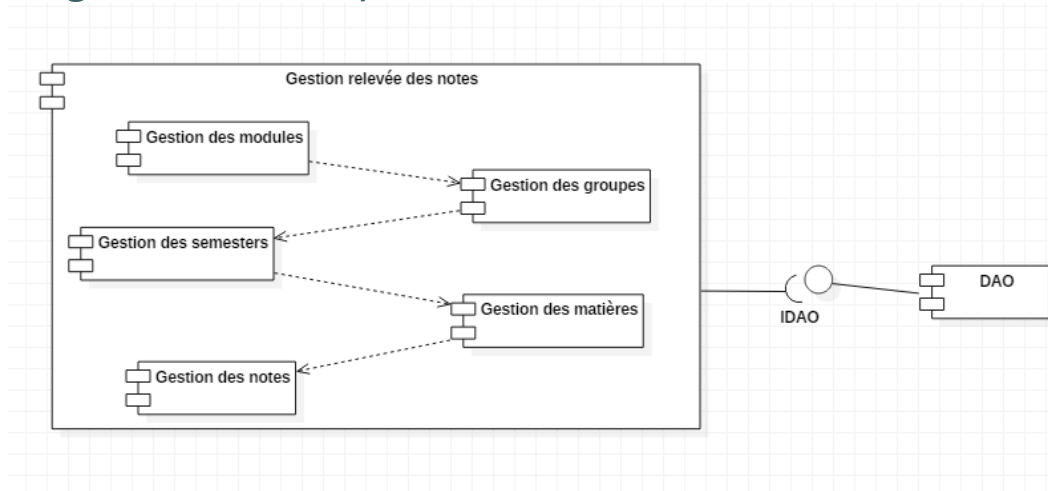


Figure 16 : diagramme de composantes



Diagramme d'états de transition :

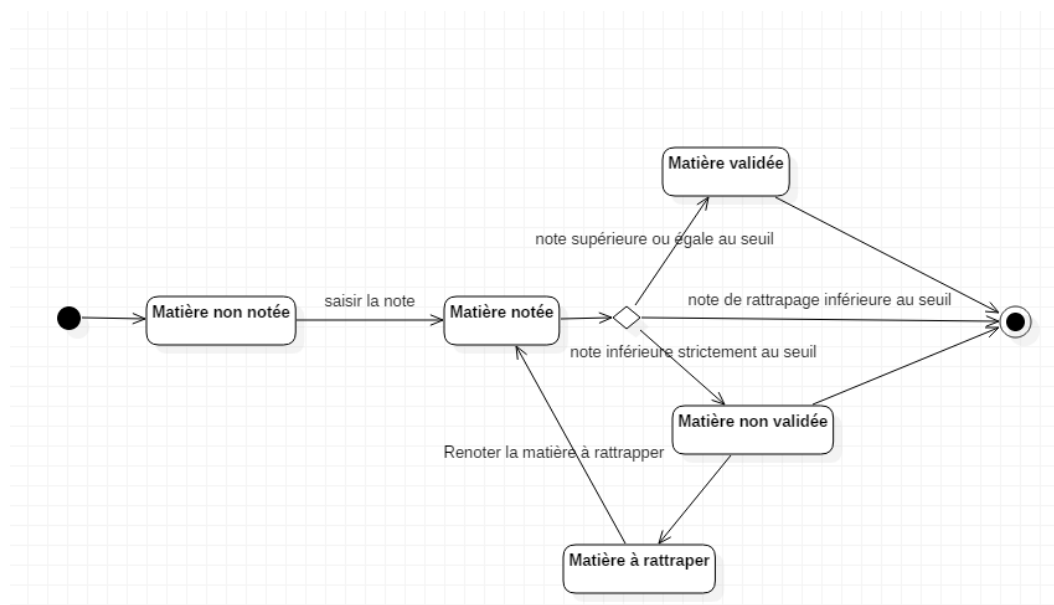


Figure 17 : Diagramme d'états de transition

Ce diagramme présente les différents états de l'objet matière, au début il est dans l'état non notée, lorsque l'agent de saisie introduit la note il passe à l'état noté, si la note est valide la matière est considérée comme validée sinon il faut la rattraper, si la note de encore inférieur du seuil, elle reste invalide.



Diagramme d'activités :

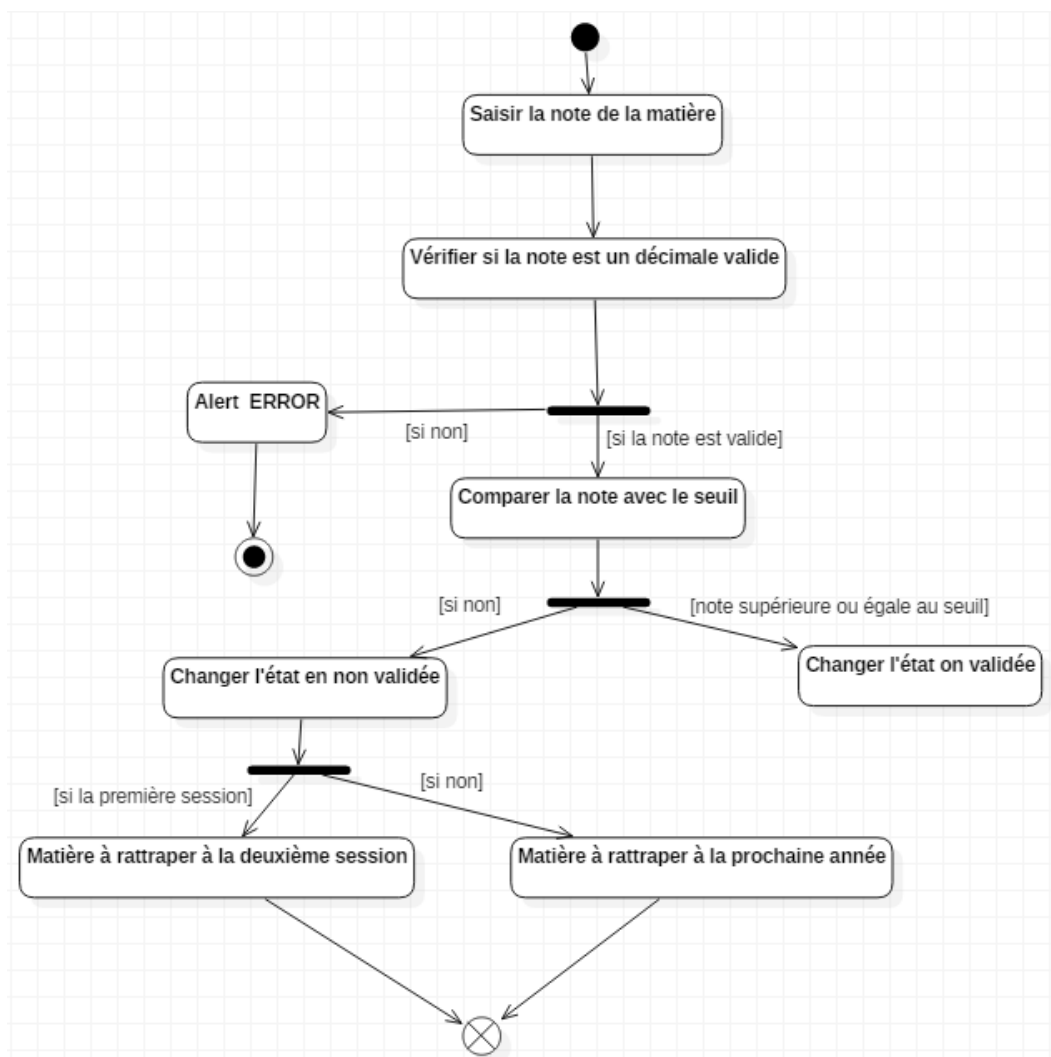


Figure 18 : Diagramme d'activités



Les diagrammes d'activités sont relativement proches des diagrammes d'états-transitions dans leur présentation, mais leur interprétation est sensiblement différente. Les diagrammes d'états-transitions sont orientés vers des systèmes réactifs, mais ils ne donnent pas une vision satisfaisante d'un traitement faisant intervenir plusieurs classeurs et doivent être complétés, par exemple, par des diagrammes de séquence. Au contraire, les diagrammes d'activités ne sont pas spécifiquement rattachés à un classeur particulier. On peut attacher un diagramme d'activités à n'importe quel élément de modélisation afin de visualiser, spécifier, construire ou documenter le comportement de cet élément.

La différence principale entre les diagrammes d'interaction et les diagrammes d'activités est que les premiers mettent l'accent sur le flot de contrôle d'un objet à l'autre, tandis que les seconds insistent sur le flot de contrôle d'une activité à l'autre.

Dans ce diagramme d'activités je donne plus de détails sur le traitement de la validation des matières en fonction de leurs notes.



Conclusion :

A la fin de ce rapport j'aimerais remercier mon professeur pour ces efforts. Ce projet est une occasion intéressante pour mettre en pratique toutes les notions du cours. JAVA est un langage très employé dans le domaine professionnel, pour ce là ce projet est un bon exercice pour améliorer mes compétences de programmation.