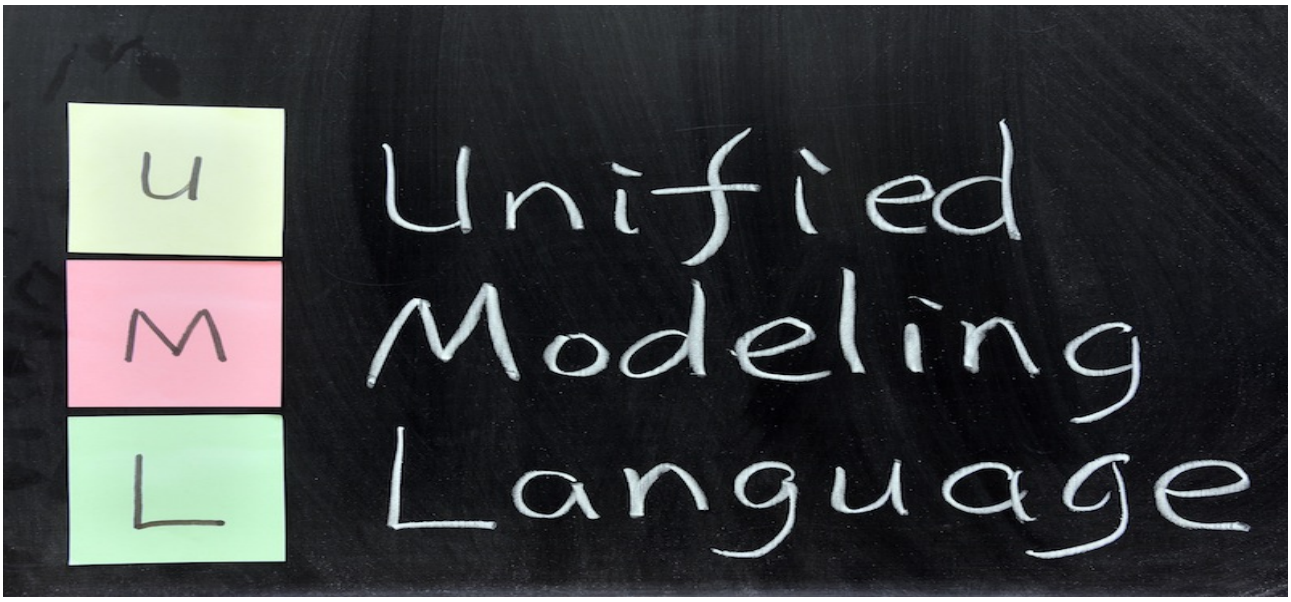
Université d'Oran 1 Ahmed Ben Bella
Module GL2 TP
Année universitaire 2017/2018

Oran, Décembre 2017

Projet GL2 TP :

Conception et modélisation sur la gestion d'une école avec UML et Java

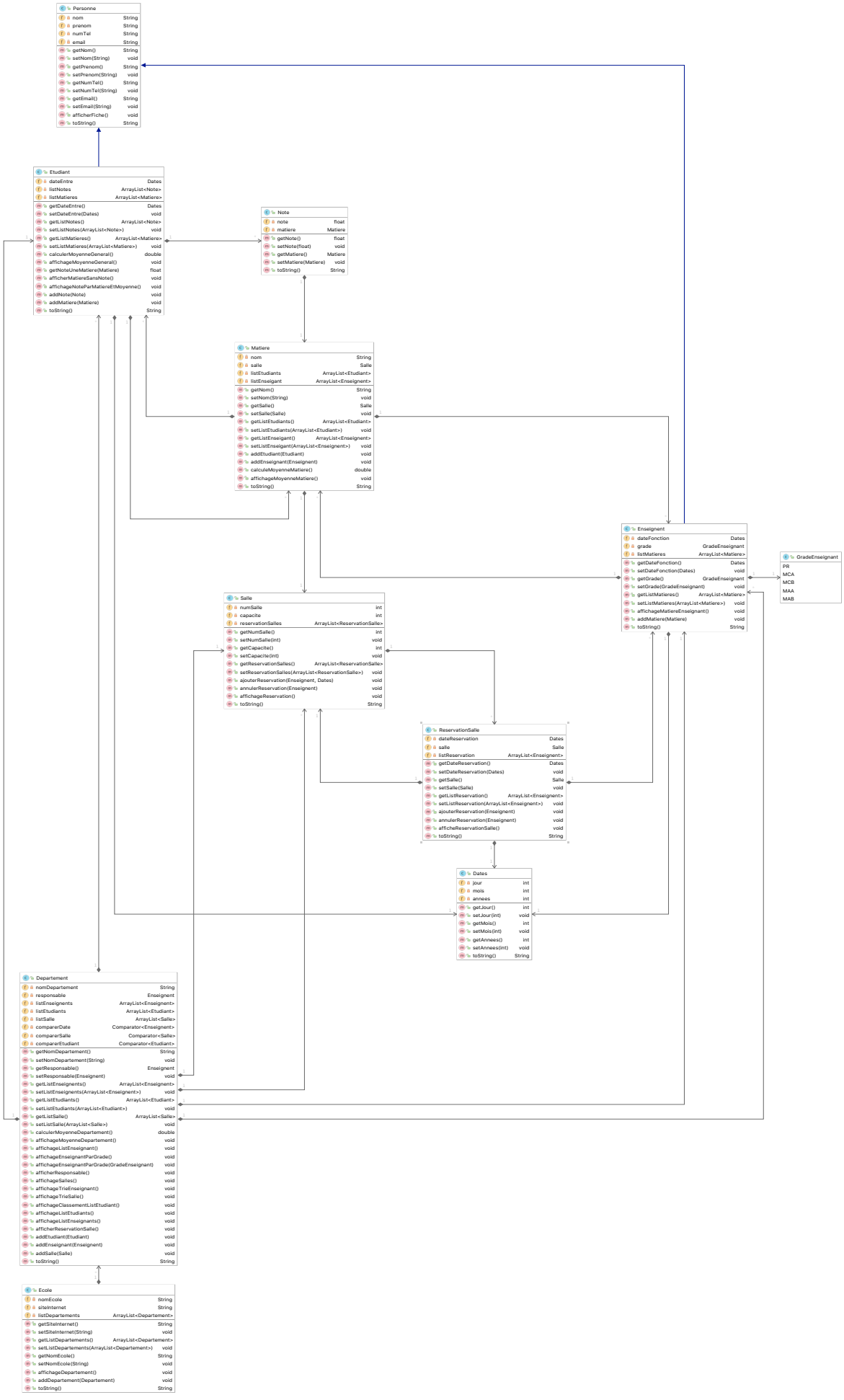


Travail réalisé par :

- CHOUIB Chawki (groupe 1)
- DJEZIRI Ibtissem (groupe 1)

Table des matières :

I. Diagramme de classe du projet



II. Réponse aux questions de l'énoncé

- Pour pouvoir répondre aux questions , on a utiliser une instance de :
  - 1 ecole.
  - 7 enseignants.
  - 2 departements.
  - 6 salles.
  - 6 matières.
  - 6 étudiants.
  - 10 reservations.

1. Calculer la moyenne par matière

```java
//calculer la moyenne des notes de la matiere
public double calculeMoyenneMatiere() {
    double moyenne = 0;
    for (int i = 0; i < listEtudiants.size(); i++) {
        moyenne = listEtudiants.get(i).getNoteUneMatiere( NomMatiere: this) + moyenne;
    }
    return (moyenne / listEtudiants.size());
}
```

- La class Matière possède une liste d'étudiants qui sont inscrit a cette matière Et dans la classe étudiant elle possède une liste de note de cet étudiant.

- Pour simplifier les choses on a rajouter une méthode qui cherche la note de cet étudiant dans la matière concerner.

```java
//get une note d'une matiere donnée
public float getNoteUneMatiere(Matiere NomMatiere) {
    for (int i = 0; i < listNotes.size(); i++) {
        if (listNotes.get(i).getMatiere().equals(NomMatiere)) {
            return listNotes.get(i).getNote();
        }
    }
    return 0;
}
```

- Donc pour calculer la moyenne d'une matière il Sufi de sommé les notes des étudiant inscrit dans cette matière puis de deviser par le nombre de ces étudiants

2. Calculer la moyenne par département

```java
//calculer la moyenne general du departement
public double calculerMoyenneDepartement() {
    double moyenne = 0;
    for (int i = 0; i < listEtudiants.size(); i++) {
        moyenne = listEtudiants.get(i).calculerMoyenneGeneral() + moyenne;
    }
    return moyenne / listEtudiants.size();
}
```

- La classe département possède une liste d'étudiants et dans la classe étudiant on a une liste de note et une méthode qui calcule la moyenne general de l'étudiant concerné.

- Donc pour calculer la moyenne du département il suffi de sommé les moyennes des étudiants du département puis de la deviser par le nombre de ces étudiants.

3. Calculer la moyenne general d'un étudiant

```java
//calculer la moyenne de l'etudiant
public double calculerMoyenneGeneral() {
    double moyenne = 0;
    for (int i = 0; i < listNotes.size(); i++) {
        moyenne = listNotes.get(i).getNote() + moyenne;
    }
    return (moyenne / listMatieres.size());
}
```

- La class étudiant possède une liste de note de l'étudiant et une méthode qui calcule la moyenne general.

- Donc pour calculer la moyenne il suffi de sommé toutes les notes de l'étudiant puis de deviser la somme par le nombre de matière dans laquelle il est inscrit.

4. Afficher la fiche Signalétique d'une personne

```java
//affichage de la fiche d'une personne
public void afficherFiche() {
    System.out.println(this);
}

@Override
public String toString() {
    return "nom='" + nom + '\'' +
            ", prenom='" + prenom + '\'' +
            ", numTel='" + numTel + '\'' +
            ", email='" + email + '\'';
}
```

- La classe personne possède la méthode d'affichage de la fiche qui marche avec la méthode ***toString().***

- *Avec l'heritage cette méthode est accessible par la classe enseignant et étudiant.*

5. Affichage des enseignants par grade

```java
//affichage des enseiagnt par un grad
public void affichageEnseignantParGrade(GradeEnseignant grade) {
    for (int i = 0; i < listEnseignents.size(); i++) {
        if (listEnseignents.get(i).getGrade().equals(grade))
            listEnseignents.get(i).afficherFiche();

    }
}
```

- La classe département possède une liste d'enseignant , donc on va parcourir cette liste avec une boucle ***'pour'*** *et on compare le grade de l'enseignant avec le grade donner en paramètre de la méthode , si identique on l'affiche , sinon on passe au suivant.*

6. Affichage des département d'une école

```java
//affichage des departement
public void affichageDepartement() {
    System.out.println("Les departements de " + nomEcole + " sont :");
    for (int i = 0; i < listDepartements.size(); i++) {
        System.out.println(i + 1 + "- " + listDepartements.get(i).getNomDepartement());
    }
}
```

- La classe école possède une liste de département , donc il suffi juste d'afficher cette liste avec une boucle ***'pour'.***

7. Affichage du chef de département

```java
//affichage du responsable du departement
public void afficherResponsable() {
    System.out.println("le responsabe du departement " + nomDepartement + " est : " + responsable.getNom());
}
```

- La classe département possède l'attribut chef de département , il suffi juste de l'afficher.

8. Affichage des cours d'un enseignant

```java
//affiche les matiere de l'enseignant
public void affichageMatiereEnseignant() {
    System.out.println("\n" + getNom() + " enseigne : ");
    for (int i = 0; i < listMatieres.size(); i++) {
        System.out.println("-la matiere " + listMatieres.get(i).getNom());
    }
}
```

*- La classe enseignant possède une liste de matière , il suffi juste d'afficher cette liste avec une boucle '****pour'.***

9. Affiche des enseignant du plus ancien au plus recent

```java
//Methode pour trier la liste des enseignant par date
private Comparator<Enseignent> comparerDate = new Comparator<Enseignent>() {
    @Override
    public int compare(Enseignent o1, Enseignent o2) {
        if (o1.getDateFonction() == o2.getDateFonction())
            return 0;
        else if (o1.getDateFonction().getAnnees() < o2.getDateFonction().getAnnees())
            return -1;
        else
            return 1;
    }
};

public void affichageTrieEnseignant() {
    listEnseignents.sort(comparerDate);
    for (int i = 0; i < listEnseignents.size(); i++) {
        System.out.println(listEnseignents.get(i));

    }
}
```

- La méthode `compare(Object o1, Object o2)` de l'interface Comparator prend deux objets.

- Dans notre exemple, elle doit être implémentée trois fois de manière qu'elle retourne un entier négative si le premier argument est plus petit que que le deuxième et zéro s'ils sont égaux et un entier positive si le premier argument est plus grand que que le deuxième.

- Une fois la méthode implémenter il suffi juste de passer en paramètre de la méthode `arrayList.sort(Comparator c)` l'objet comparator qu'on a implémenté et la liste serra trier en fonction de la date de chaque enseignant par ordre croissant et ensuite on affiche cette liste avec une boucle ***'pour'.***

10. Affichage des salle par ordre décroissant de leur capacités

```java
// Methode pour trier la liste des salles par capacité
private Comparator<Salle> comparerSalle = new Comparator<Salle>() {
    @Override
    public int compare(Salle o1, Salle o2) {
        if (o1.getCapacite() == o2.getCapacite())
            return 0;
        else if (o1.getCapacite() > o2.getCapacite())
            return -1;
        else
            return 1;
    }
};

public void affichageTrieSalle() {
    listSalle.sort(comparerSalle);
    for (int i = 0; i < listSalle.size(); i++) {
        System.out.println(listSalle.get(i));

    }
}
```

- La méthode `compare(Object o1, Object o2)` de l'interface Comparator prend deux objets.

- Dans notre exemple, elle doit être implémentée trois fois de manière qu'elle retourne un entier négative si le premier argument est plus grand que que le deuxième et zéro s'ils sont égaux et un entier positive si le premier argument est plus petit que que le deuxième.

- Une fois la méthode implémenter il suffi juste de passer en paramètre de la méthode `arrayList.sort(Comparator c)` l'objet comparator qu'on a implémenté et la liste serra trier en fonction de la capacité de chaque salle par ordre décroissant et ensuite on affiche cette liste avec une boucle ***'pour'.***

11. Affichage des notes par matière et la moyenne d'un étudiant

```java
//affichage des listNotes et moyenne general
public void affichageNoteParMatiereEtMoyenne() {
    System.out.println("\nEtudiant " + getNom() + " " + getPrenom());
    for (int i = 0; i < listNotes.size(); i++) {
        System.out.println("la note de la matiere " + listNotes.get(i).getMatiere().getNom() + " est " + listNotes.get(i).getNote());
    }
    afficherMatiereSansNote();
    affichageMoyenneGeneral();
}
```

- La classe étudiant possède une liste de note , il suffi juste de les afficher avec une boucle ***'pour'*****.**

```java
//liste des matiere sans note de l'etudiant
public void afficherMatiereSansNote() {
    ArrayList<Matiere> matiereSansNote = new ArrayList<>();

    for (int i = 0; i < listMatieres.size(); i++) {

        int indice = 0;
        boolean existe = false;

        while (!existe && indice < listNotes.size()) {
            if (listMatieres.get(i).equals(listNotes.get(indice).getMatiere())) {
                existe = true;
            } else
                indice++;
        }

        if (!existe) {
            matiereSansNote.add(listMatieres.get(i));
        }
    }
    if (matiereSansNote.isEmpty())
        System.out.println("l'étudiant " + getNom() + " est noté sur toutes les Matieres");
    else {
        System.out.println("les Matieres sans note de " + getNom());
        for (int i = 0; i < matiereSansNote.size(); i++) {
            System.out.println(matiereSansNote.get(i).getNom());
        }
    }
}
```

- Pour afficher les matières sans note il suffi de comparer les notes dans la liste note de l'étudiant et les matière dans la liste matière de l'étudiant , on utilise l'intersection des deux liste.

- Si intersection vide alors on affiche un message sinon on affiche la liste de ces matière sans note.

- La méthode pour afficher la moyenne est deja défini dans la ***page n°4.***

12. Affichage du classement des étudiants par moyenne

```
// Methode pour trier la liste des Etudiants par classement
private Comparator<Etudiant> comparerEtudiant = new Comparator<Etudiant>() {
    @Override
    public int compare(Etudiant o1, Etudiant o2) {
        if (o1.calculerMoyenneGeneral() == o2.calculerMoyenneGeneral())
            return 0;
        else if (o1.calculerMoyenneGeneral() > o2.calculerMoyenneGeneral())
            return -1;
        else
            return 1;
    }
};

public void affichageClassementListEtudiant() {
    listEtudiants.sort(comparerEtudiant);
    for (int i = 0; i < listEtudiants.size(); i++) {
        System.out.println(listEtudiants.get(i) + " moyenne : " + listEtudiants.get(i).calculerMoyenneGeneral());

    }
}
```

- La méthode `compare(Object o1, Object o2)` de l'interface Comparator prend deux objets.

- Dans notre exemple, elle doit être implémentée trois fois de manière qu'elle retourne un entier négative si le premier argument est plus grand que que le deuxième et zéro s'ils sont égaux et un entier positive si le premier argument est plus petit que que le deuxième.

*- Une fois la méthode implémenter il suffi juste de passer en paramètre de la méthode `arrayList.sort(Comparator c)` l'objet comparator qu'on a implémenté et la liste serra trier en fonction de la moyenne de chaque étudiant par ordre décroissant et ensuite on affiche cette liste avec une boucle **'pour'.***

13. Affichage des matière sans note d'un étudiant

- La méthode pour afficher les matière sans note est deja défini dans la ***page n°9.***

14. Reservation d'une salle

```java
//methode de rervation d'une salle par un enseignant
public void ajouterReservation(Enseignent enseignent, Dates datesReservation) {
    boolean reserver = false;
    for (int i = 0; i < reservationSalles.size(); i++) {
        if (reservationSalles.get(i).getDateReservation().getJour() == (datesReservation.getJour())
                && reservationSalles.get(i).getDateReservation().getMois() == (datesReservation.getMois())
                && reservationSalles.get(i).getDateReservation().getAnnees() == (datesReservation.getAnnees())) {
            reservationSalles.get(i).ajouterReservation(enseignent);
            reserver = true;
            break;
        }
    }
    if (!reserver) {
        reservationSalles.add(new ReservationSalle(datesReservation, salle: this, enseignent));
    }
}
```

- La classe Salle possède une liste de reservation (type reservation).
- On compare la date de reservation donner en paramètre de la méthode.
- si cette date existe deja dans la liste alors reserve cette salle a la meme date.
- sinon on cree une nouvelle réservation a cette date donner en paramètre.

```java
//ajouter reservation d'une salle par un enseignant
public void ajouterReservation(Enseignent enseignent) {
    if (listReservation.size() < 5) {
        listReservation.add(enseignent);
    } else
        System.out.println("impossible de reserver cette salle (complet!)");
}
```

- La clase ReservationSalle poseed une file de reservation de taille maximum 5.
- Il est impossible pour un enseignant de reserver une salle qui est deja reserver au plus 5 fois dans la meme journée.

15. Annuler une reservation

```java
//annuler une reservation d'un enseignant
public void annulerReservation(Enseignent enseignent) {
    for (int i = 0; i < reservationSalles.size(); i++) {
        reservationSalles.get(i).annulerReservation(enseignent);
    }
}
```

- *La classe Salle possède une liste de reservation (type reservation).*
- *Pour annuler une reservation on parcourt la liste de reservation faite par un enseignant et en supprime l'objet de la liste , le décalage se ferra automatique par la méthode.*

```java
//annuler une reservation d'un enseignant
public void annulerReservation(Enseignent enseignent) {
    if (!listReservation.isEmpty()) {
        listReservation.remove(enseignent);
    } else {
        System.out.println("pas de reservation pour cet enseignant " + enseignent.getNom());
    }
}
```

- *La clase ReservationSalle poseed une file de reservation de taille maximum 5.*
- *On utilisant la méthode* `array.remove(Enseignant object)` *et on supprime toute les reservation de cette enseignant dans cette salle du meme jour.*