

# POO - Labo 6

- Groupe: L06GrJ
- Etudiants: Calum Quinn, Dylan Ramos
- Date de dernière modification: 05.11.2023

## Choix d'implémentation

### Classe `Etudiant`

Nous avons choisi d'ajouter un attribut privé `groupe` et une méthode publique `definirGroupe()` à la classe `Etudiant`.

En effet, pour pouvoir afficher l'étudiant lors d'un appel `toString()`, il faut savoir à quel groupe il appartient afin d'obtenir un affichage similaire à celui de l'énoncé.

### Classe `Professeur`

Nous avons choisi d'ajouter un tableau de leçons en tant qu'attribut privé et une méthode publique `definirLecons()` à

la classe `Professeur`. En effet, pour pouvoir afficher l'horaire d'un professeur, il faut savoir à quelles leçons il participe.

### Classe `Groupe`

Nous avons choisi d'ajouter un tableau d'étudiants en tant qu'attribut privé à la classe `Groupe` qui sera rempli à l'aide du constructeur lors de la création d'un groupe. Le constructeur vérifie qu'il y ait au moins 1 étudiant (pour respecter la cardinalité de la relation) et itère sur chaque étudiant pour leur assigner le groupe.

Enfin, comme pour la classe `Professeur`, nous avons ajouté un tableau de leçons en tant qu'attribut privé afin de stocker les leçons auxquelles le groupe participe. Cela s'avère utile pour l'affichage de l'horaire.

### Classe `Lecon`

Etant donné que la méthode `horaire()` de la classe `Lecon` est statique, nous avons implémenté la méthode `horaire()`

des classes `Professeur` et `Groupe` de manière à ce qu'ils fassent appel à la méthode `horaire()` de la classe `Lecon`.

En effet, l'affichage de l'horaire a toujours le même format que ce soit pour un professeur ou un groupe, il est donc

plus adapté de factoriser le code à un seul endroit. De plus, nous stockons les leçons dans les classes

`Professeur`

et `Groupe`, ce qui nous permet de les passer en paramètre à la méthode `horaire()` de la classe `Lecon`.

Pour la création de l'horaire, nous avons choisi d'utiliser un tableau de `String` à deux dimensions afin de pouvoir créer les cellules colonne par colonne. En effet, créer les cellules de cette manière permet de simplifier le code lors

de la création d'une cellule de plusieurs périodes. Enfin, une fois le tableau rempli, nous concaténons les colonnes

afin d'obtenir l'affichage souhaité.

# Folder src

6 printable files

(file list disabled)

src\Etudiant.java

```
package src;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Etudiant extends Personne {
    private final int matricule;
    private Groupe groupe;

    public Etudiant(String nom, String prenom, int matricule) {
        super(nom, prenom);
        if (matricule < 0) {
            throw new RuntimeException("Le numéro de matricule doit être positif");
        }
        this.matricule = matricule;
    }

    public String toString() {
        return String.format("Etud. %s (%s) - %s", super.toString(), matricule, groupe.nom());
    }

    public void definirGroupe(Groupe groupe) {
        this.groupe = groupe;
    }
}
```

src\Groupe.java

```
package src;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Groupe {
    private final int numero;
    private final int trimestre;
    private final String orientation;
    private final Etudiant[] etudiants;
    private Lecon[] lecons;

    public Groupe(int numero, int trimestre, String orientation, Etudiant[] etudiants) {
        if (etudiants == null) {
            throw new RuntimeException("Il faut au moins 1 étudiant dans un groupe");
        }
        if(numero < 0 || trimestre < 0 || orientation == null || orientation.isEmpty()) {
            throw new RuntimeException("Le numéro et le trimestre doivent être positifs et l'orientation ne peut pas être vide");
        }
        this.numero = numero;
        this.trimestre = trimestre;
        this.orientation = orientation;
        this.etudiants = etudiants;
        // Assign group to each student
        for (Etudiant etudiant : etudiants) {
            etudiant.definirGroupe(this);
        }
    }
}
```

```

    }

}

    public String horaire() {
        return String.format("\n-- Horaire du groupe %s (%s étudiants)\n\n%s", nom(), nombreEtudiants(),
Lecon.horaire(lecons));
    }

    public String nom() {
        return String.format("%s%d-%d", orientation, trimestre, numero);
    }

    public int nombreEtudiants() {
        return etudiants.length;
    }

    public void definirLecons(Lecon[] lecons) {
        this.lecons = lecons;
    }
}

```

src\Lecon.java

```

package src;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Lecon {
    private final String matiere;
    private final int jourSemaine;
    private final int periodeDebut;
    private final int duree;
    private final String salle;
    private final Professeur professeur;
    private static final char caractereColonne = '|';
    private static final int largeurCellule = 13;

    public Lecon(String matiere, int jourSemaine, int periodeDebut, int duree, String salle, Professeur professeur) {
        if (matiere == null || matiere.isEmpty()) {
            throw new RuntimeException("La matière ne peut pas être vide");
        }
        if (jourSemaine < 1 || jourSemaine > 5) {
            throw new RuntimeException("Le jour de la semaine doit être entre 1 et 5");
        }
        if (periodeDebut < 1 || periodeDebut > 11) {
            throw new RuntimeException("La période de début doit être entre 1 et 11");
        }
        if (duree % 45 != 0 || duree > 495) {
            throw new RuntimeException("La durée doit être un multiple de 45 et inférieure à 495");
        }
        if (salle == null || salle.isEmpty()) {
            throw new RuntimeException("La salle ne peut pas être vide");
        }
        this.matiere = matiere;
        this.jourSemaine = jourSemaine;
        this.periodeDebut = periodeDebut;
        this.duree = duree;
        this.salle = salle;
        this.professeur = professeur;
    }

    public static String horaire(Lecon[] lecons) {
        // Create a 2D array to store the schedule
        final int nbLignes = 24, nbColonnes = 6;
        String[][] grille = new String[nbLignes][nbColonnes];

        // Create the left and top headers
        creerEntetes(grille);
    }
}

```

```

// Fill the schedule column by column to simplify the code for multiple lessons in the same day
for (int i = 1; i < nbColonnes; ++i) {
    int nbSeparationsVides = 0;

    for (int j = 2; j < nbLignes; ++j) {
        // If the row is even, it's a lesson else it's a separation
        if (j % 2 == 0) {
            boolean leconTrouvee = false;
            // Create a lesson cell if a lesson was found for the current day and period
            for (Lecon lecon : lecons) {
                if (lecon.jourSemaine == i && lecon.periodeDebut == j / 2) {
                    String professeurAbreviation = lecon.professeur != null ? lecon.professeur.abreviation() :
";

                    grille[j][i] = cellule(lecon.matiere, lecon.salle, professeurAbreviation);
                    // Calculate the number of empty separations needed for the lesson
                    nbSeparationsVides = lecon.duree / 45 - 1;
                    leconTrouvee = true;
                    break;
                }
            }
            // If no lesson was found, we create an empty cell
            if (!leconTrouvee) {
                grille[j][i] = cellule(null, null, null);
            }
        } else {
            // Create an empty separation cell or a cell with a separation character
            if (nbSeparationsVides > 0) {
                grille[j][i] = celluleSeparation(true);
                --nbSeparationsVides;
            } else {
                grille[j][i] = celluleSeparation(false);
            }
        }
    }
}

// Create the schedule string by concatenating the columns
StringBuilder horaire = new StringBuilder();

for (int i = 0; i < nbLignes; ++i) {
    for (int j = 0; j < nbColonnes; ++j) {
        horaire.append(grille[i][j]);
    }
    horaire.append("\n");
}

return horaire.toString();
}

private static void creerEntetes(String[][] grille) {
    // Top header
    final String[] jours = {"Lun", "Mar", "Mer", "Jeu", "Ven"};
    for (int i = 1; i < grille[0].length; ++i) {
        // We need to decrease the width by 1 because of the first space
        grille[0][i] = String.format("%-" + (largeurCellule - 1) + "s", jours[i - 1], caractereColonne);
        grille[1][i] = celluleSeparation(false);
    }

    // Left header
    final String[] heures = {"8:30", "9:15", "10:25", "11:15", "12:00", "13:15", "14:00", "14:55", "15:45",
        "16:35", "17:20"};
    grille[0][0] = celluleHeure(null);
    grille[1][0] = celluleHeure(null);
    for (int i = 2; i < grille.length; ++i) {
        if (i % 2 == 0) {
            grille[i][0] = celluleHeure(heures[i / 2 - 1]);
        } else {
            grille[i][0] = celluleHeure(null);
        }
    }
}
}

```

```

private static String celluleHeure(String heure) {
    final int largeur = 5;
    heure = heure != null ? heure : " ";

    return String.format("%" + largeur + "s", heure, caractereColonne);
}

private static String cellule(String matiere, String salle, String professeur) {
    matiere = matiere != null ? matiere : "";
    salle = salle != null ? salle : "";
    String professeurAbreviation = professeur != null ? professeur : "";
    String texteCellule = String.format("%s%3s%s %s", matiere, " ", salle, professeurAbreviation);

    return String.format("%-" + largeurCellule + "s", texteCellule, caractereColonne);
}

private static String celluleSeparation(boolean estVide) {
    char caractereSeparation = estVide ? ' ' : '-';

    return String.format("%" + largeurCellule + "s", " ").replace(' ', caractereSeparation) + caractereColonne;
}
}

```

src/Main.java

```

package src;

import java.util.Arrays;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Main {
    public static void main(String[] args) {
        Professeur[] professeurs = new Professeur[]{
            new Professeur("Rossier", "Daniel", "DRE"),
            new Professeur("Donini", "Pier", "PDO")
        };

        Lecon[] lecons = new Lecon[]{
            new Lecon("PO0", 3, 6, 90, "H02", professeurs[1]),
            new Lecon("PO0", 4, 6, 90, "H02", professeurs[1]),
            new Lecon("PO0", 4, 8, 90, "H02", professeurs[1]),
            new Lecon("SYE", 1, 1, 90, "G01", professeurs[0]),
            new Lecon("SYE", 4, 3, 90, "A09", professeurs[0]),
            new Lecon("TIC", 2, 10, 45, "F06", null)
        };

        Etudiant[] etudiants = new Etudiant[]{
            new Etudiant("Lennon", "John", 1234),
            new Etudiant("Mc Cartney", "Paul", 2341),
            new Etudiant("Starr", "Ringo", 3241),
            new Etudiant("Harrison", "George", 4321),
            new Etudiant("Waters", "Roger", 1324),
            new Etudiant("Gilmour", "David", 4312),
        };

        Groupe[] groupes = new Groupe[]{
            new Groupe(1, 6, "IL", Arrays.copyOfRange(etudiants, 0, 4)),
            new Groupe(1, 6, "SI", Arrays.copyOfRange(etudiants, etudiants.length - 2, etudiants.length))
        };

        // Display school's members
        System.out.println("-- Membres de l'ecole\n");
        Personne[] personnes = new Personne[professeurs.length + etudiants.length];
        System.arraycopy(professeurs, 0, personnes, 0, professeurs.length);
        System.arraycopy(etudiants, 0, personnes, professeurs.length, etudiants.length);
    }
}

```

```

        for (Personne personne : personnes) {
            System.out.println(personne);
        }

        // Define lessons for the group IL6-1
        groupes[0].definirLecons(lecons);
        // Define lessons for the teacher PDO
        professeurs[1].definirLecons(Arrays.copyOfRange(lecons, 0, 3));

        System.out.println(groupe[0].horaire());
        System.out.println(professeurs[1].horaire());
    }
}

```

src\Personne.java

```

package src;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public abstract class Personne {
    private final String nom;
    private final String prenom;

    public Personne(String nom, String prenom) {
        if (nom == null || prenom == null || nom.isEmpty() || prenom.isEmpty()) {
            throw new RuntimeException("Le nom et le prénom ne peuvent pas être vides");
        }
        this.nom = nom;
        this.prenom = prenom;
    }

    public String toString() {
        return String.format("%s %s", prenom, nom);
    }
}

```

src\Professeur.java

```

package src;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Professeur extends Personne {
    private final String abreviation;
    private Lecon[] lecons;

    public Professeur(String nom, String prenom, String abreviation) {
        super(nom, prenom);
        if (abreviation == null || abreviation.isEmpty()) {
            throw new RuntimeException("L'abréviation ne peut pas être vide");
        }
        this.abreviation = abreviation;
    }

    public String abreviation() {
        return abreviation;
    }

    public String horaire() {
        return String.format("\n-- Horaire %s\n%s", this, Lecon.horaire(lecons));
    }

    public String toString() {

```

```
        return String.format("Prof. %s (%s)", super.toString(), abbreviation);
    }

    public void definirLecons(Lecon[] lecons) {
        this.lecons = lecons;
    }
}
```