```java
package ch.heig.poo.labo6;

import ch.heig.poo.labo6.école.*;
import java.util.Arrays;

/**
 * Classe permettant de tester les fonctionnalités demandées dans la donnée du laboratoire 6
 * Date de création 17.11.2022
 *
 * @author Kevin Farine, Timothée Van Hove
 */
public class Main {
    public static void main(String[] args) {

        //1.
        Professeur dre = new Professeur("Rossier", "Daniel", "DRE");
        Professeur pdo = new Professeur("Donini", "Pier", "PDO");

        //2.
        Leçon poo1 = new Leçon("POO", 3, 6, 2, "H02", pdo);
        Leçon poo2 = new Leçon("POO", 4, 6, 2, "H02", pdo);
        Leçon poo3 = new Leçon("POO", 4, 8, 2, "H02", pdo);
        Leçon sye1 = new Leçon("SYE", 1, 1, 2, "G01", dre);
        Leçon sye2 = new Leçon("SYE", 4, 3, 2, "A09", dre);
        Leçon tic1 = new Leçon("TIC", 2, 10, 1, "F06");

        //3.
        Etudiant john = new Etudiant("Lennon", "John", 1234);
        Etudiant paul = new Etudiant("Mc Cartney", "Paul", 2341);
        Etudiant ringo = new Etudiant("Starr", "Ringo", 3241);
        Etudiant george = new Etudiant("Harisson", "George", 4321);
        Etudiant roger = new Etudiant("Waters", "Roger", 1324);
        Etudiant david = new Etudiant("Gilmour", "David", 4312);

        //4.
        Groupe il61 = new Groupe(1, "IL", 6, Arrays.asList(john, paul, ringo, george));
        Groupe si61 = new Groupe(1, "SI", 6, Arrays.asList(roger, david));

        //5.
        il61.définirLeçons(Arrays.asList(poo1, poo2, poo3, sye1, sye2, tic1));
        si61.définirLeçons(Arrays.asList(poo1, poo2, poo3));

        //6.
        System.out.println("--Membres de l'école");
        for (Personne p : new Personne[]{pdo, dre, john, paul, ringo, george, roger, david}){
            System.out.println(p);
        }
        System.out.println();

        //7.
        System.out.println(il61.horaire());

        //8.
        System.out.println(pdo.horaire());
    }
}
```

```java
package ch.heig.poo.labo6.école;

/**
 * Représente une Personne comportant un nom et un prénom
 * Date de création 17.11.2022
 *
 * @author Kevin Farine, Timothée Van Hove
 */
public class Personne {

    private final String nom;

    private final String prénom;

    /**
     * Constructeur prenant un nom et un prénom
     *
     * @param nom Le nom de famille de la personne
     * @param prénom Le prénom de la personne
     */
    public Personne(String nom, String prénom) {
        this.nom = nom;
        this.prénom = prénom;
    }

    /**
     * Convertit et retourne un String représentant les attributs d'une personne
     *
     * @return Un String représentant les attributs d'une personne
     */
    @Override
    public String toString() {
        return prénom + " " + nom;
    }
}
```

```java
package ch.heig.poo.labo6.école;

import java.util.ArrayList;

/**
 * Représente un professeur avec un nom, prénom et une abréviation. Hérite de la classe Personne
 * Date de création 17.11.2022
 *
 * @author Kevin Farine, Timothée Van Hove
 */
public class Professeur extends Personne {

    private final String abréviation;

    private final ArrayList<Leçon> leçons = new ArrayList<>();

    static final int NBRE_INITIALES = 3;

    /**
     * Constructeur prenant un nom, un prénom et une abréviation
     * @param nom Le nom de famille du professeur
     * @param prénom Le prénom du professeur
     * @param abreviation L'acronyme du professeur (par défaut, maximum 3 caractères)
     */
    public Professeur(String nom, String prénom, String abreviation) {
        super(nom, prénom);
        this.abréviation = abreviation;
    }

    /**
     * Ajoute une leçon à la liste de leçons enseignées du professeur
     * @param leçon leçon à ajouter à la liste
     */
    void définirLeçon(Leçon leçon) {
        leçons.add(leçon);
    }

    /**
     * retourne l'abréviation du professeur
     * @return L'abréviation du professeur en format String
     */
    public String abréviation() {
        return abréviation;
    }

    /**
     * Convertit et retourne les informations liées au professeur en format String
     * @return Un String représentant les informations liées au professeur
     */
    @Override
    public String toString() {
        return "Prof. " + super.toString() + " (" + abréviation + ")";
    }

    /**
     * Retourne la grille horaire du professeur en se basant sur les cours enseignés
     * @return Un String représentant la grille horaire
     */
    public String horaire() {
        return "-- Horaire " + this + "\n" + Leçon.horaire(leçons);
    }
}
```

```java
package ch.heig.poo.labo6.école;

/**
 * Représente un étudiant avec un nom, prénom et un matricule. Hérite de la classe Personne
 * Date de création 17.11.2022
 *
 * @author Kevin Farine, Timothée Van Hove
 */
public class Etudiant extends Personne {

    private final int matricule;

    private Groupe groupe;

    /**
     * Constructeur prenant un nom, un prénom et un matricule
     *
     * @param nom Le nom de l'étudiant
     * @param prénom Le prénom de l'étudiant
     * @param matricule Le matricule de l'étudiant
     * @throws RuntimeException si le matricule est < 0
     */
    public Etudiant(String nom, String prénom, int matricule) {
        super(nom, prénom);
        if (matricule < 0) {
            throw new RuntimeException("Le matricule ne peut pas être < 0");
        }
        this.matricule = matricule;
    }

    /**
     * Met à jour le groupe auquel l'étudiant est associé.
     *
     * @param groupe Le groupe dans lequel l'étudiant se trouve
     */
    void setGroupe(Groupe groupe) {
        this.groupe = groupe;
    }

    /**
     * Convertit et retourne les informations de l'étudiant
     *
     * @return String représentant les attributs de l'étudiant
     */
    @Override
    public String toString() {
        return "Etud. " + super.toString() + " (#" + matricule + ") - " + groupe.nom();
    }
}
```

```java
package ch.heig.poo.labo6.école;

import java.util.ArrayList;
import java.util.Collection;

/**
 * Représente un Groupe d'étudiants avec un nom de groupe, une orientation, d'un trimestre et un
 * numéro.
 * Date de création 17.11.2022
 *
 * @author Kevin Farine, Timothée Van Hove
 */
public class Groupe {

    private final int numéro;

    private final String orientation;

    private final int trimestre;

    private final ArrayList<Etudiant> étudiants;

    private ArrayList<Leçon> leçons;

    /**
     * Constructeur prenant un numéro, une orientation et un trimestre
     *
     * @param numéro Le numéro de groupe
     * @param orientation L'orientation (filière) du groupe
     * @param trimestre Le trimestre auquel il participe
     */
    public Groupe(int numéro, String orientation, int trimestre, Collection<Etudiant> étudiants) {
        this.numéro = numéro;
        this.orientation = orientation;
        this.trimestre = trimestre;
        this.étudiants = new ArrayList<>();
        this.étudiants.addAll(étudiants);
        leçons = new ArrayList<>();
        for (Etudiant étudiant : this.étudiants) {
            étudiant.setGroupe(this);
        }
    }

    /**
     * Ajoute une leçon à la liste de leçons du groupe
     *
     * @param leçons La liste de leçons à ajouter
     */
    public void définirLeçons(Collection<Leçon> leçons) {
        this.leçons.addAll(leçons);
    }

    /**
     * Retourne la grille horaire du groupe en se basant sur les cours suivis.
     *
     * @return Un String représentant la grille horaire du groupe
     */
    public String horaire() {
        return "-- Horaire du groupe " + nom() + " (" + nombreEtudiants() + " étudiants)\n" +
                Leçon.horaire(leçons);
    }

    /**
```

```
     * Retourne le nom du groupe, formaté à partir de l'orientation, du trimestre et du numéro
     *
     * @return Un String représentant le nom formaté du groupe
     */
    public String nom() {
        return orientation + trimestre + "-" + numéro;
    }

    /**
     * Retourne le nombre d'étudiants faisant partis du groupe
     *
     * @return Le nombre d'étudiants du groupe
     */
    public int nombreEtudiants() {
        return étudiants.size();
    }

}
```

```java
package ch.heig.poo.labo6.école;

import java.util.Collection;

/**
 * Représente une leçon contenant une matière, une salle, un jour de la semaine, une durée ainsi
 * que le numéro de la période du jour où ce cours est suivi / enseigné

 * Date de création 17.11.2022
 * Remarques : Les séparateurs de ligne / colonnes, Le nombre de jours dans la semaine, les
 * plages horaire, ainsi que la longueur des initiales peuvent être changés. L'affichage de
 * l'horaire s'adaptera automatiquement
 * Date de création 17.11.2022
 *
 * @author Kevin Farine, Timothée Van Hove
 */
public class Leçon {
    private enum Jours {Lun, Mar, Mer, Jeu, Ven}

    private static 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"};

    private static final char SEP_COL = '|', SEP_LIGNE = '-';

    private static final int LARGEUR_PREMIERE_COL = 5,
            INITIALES_MATIERE = 5, INITIALES_SALLE = 3,
            LARGEUR_COL_JOURS = 2 + Professeur.NBRE_INITIALES + INITIALES_MATIERE + INITIALES_SALLE;

    private static final String BAS_CELLULE = SEP_COL + (SEP_LIGNE + "").repeat(LARGEUR_COL_JOURS),
            CELLULE_VIDE = BAS_CELLULE.replace(SEP_LIGNE, ' '),
            SEP_HEURE = " ".repeat(LARGEUR_PREMIERE_COL),
            SEP_LIGNE_COMPLETE = SEP_HEURE + BAS_CELLULE.repeat(Jours.values().length) + SEP_COL +
            "\n",
            FORMAT_HEURE = "%" + LARGEUR_PREMIERE_COL + "s",
            FORMAT_CELLULE = SEP_COL + "%-" + INITIALES_MATIERE + "s %" + INITIALES_SALLE + "s %" +
                    Professeur.NBRE_INITIALES + "s";

    private final String matière;

    private final int jourSemaine;

    private final int périodeDébut;

    private final int durée;

    private final String salle;

    private Professeur professeur;

    /**
     * Constructeur de la Leçon
     * @param matière L'acronyme de la matière de la leçon (par défaut 5 caractères maximum)
     * @param jourSemaine Le jour de la semaine 1 (lundi) à 5 (vendredi)
     * @param périodeDébut Le numéro de période à laquelle la leçon commence (1 - 11 par défaut)
     * @param durée La durée (en périodes) de la leçon
     * @param salle L'acronyme de la salle de cours. Par défaut, maximum 3 caractères
     * @throws RuntimeException Si durée, périodeDébut sont inférieurs à 0, si la durée de la
     * leçon dépasse les plages horaires de la semaine, si jourSemaine est hors plage, ou si la
     * longueur des initiales est trop grande.
     */
    public Leçon(String matière, int jourSemaine, int périodeDébut, int durée, String salle) {
        if (périodeDébut + durée > HEURES.length) {
            throw new RuntimeException("La durée de la leçon excède la longueur de la plage " +
```

```
                    "horaire");
        }
        if (durée < 1 || périodeDébut < 1) {
            throw new RuntimeException("La durée et la période de début doivent être > 0");
        }
        if (jourSemaine < 1 || jourSemaine > Jours.values().length) {
            throw new RuntimeException("Le jour de la semaine doit être entre 1 et " +
                    Jours.values().length);
        }
        if (salle.length() > INITIALES_SALLE) {
            throw new RuntimeException("Les initiales de la salle ne peuvent pas dépasser" +
                    INITIALES_SALLE + " caractères. Actuellement : " + salle.length());
        }
        if (matière.length() > INITIALES_MATIERE) {
            throw new RuntimeException("Les initiales de la matière ne peuvent pas dépasser" +
                    INITIALES_MATIERE + " caractères. Actuellement : " + matière.length());
        }

        this.matière = matière;
        this.jourSemaine = jourSemaine;
        this.périodeDébut = périodeDébut;
        this.durée = durée;
        this.salle = salle;
    }

    /**
     * Constructeur de la Leçon
     * @param matière L'acronyme de la matière de la leçon (par défaut 5 caractères maximum)
     * @param jourSemaine Le jour de la semaine 1 (lundi) à 5 (vendredi)
     * @param périodeDébut Le numéro de période à laquelle la leçon commence (1 - 11 par défaut)
     * @param durée La durée (en périodes) de la leçon
     * @param salle L'acronyme de la salle de cours. Par défaut, maximum 3 caractères
     * @param professeur Le professeur assigné à cette leçon
     */
    public Leçon(String matière, int jourSemaine, int périodeDébut, int durée, String salle,
                 Professeur professeur) {
        this(matière, jourSemaine, périodeDébut, durée, salle);
        if (this.professeur != null) {
            this.professeur = professeur;
            this.professeur.définirLeçon(this);
        }
    }

    /**
     * Construit l'en-tête de la grille horaire
     * @return l'en-tête de la grille horaire
     */
    private static StringBuilder CréerEnTête() {
        StringBuilder enTête = new StringBuilder(SEP_HEURE);

        //Ajouter tous les jours de la semaine
        for (Jours jour : Jours.values()) {
            enTête.append(String.format(SEP_COL + " %-" + (LARGEUR_COL_JOURS - 1) + "s",
                    jour.name()));
        }
        //Ajouter le dernier séparateur, le retour à la ligne et le bas des cellules de l'en-tête
        return enTête.append(SEP_COL + "\n").append(SEP_LIGNE_COMPLETE);
    }

    /**
     * Construit la cellule de la grille horaire en fonction des lignes paires (contenu de la
     * cellule) et des lignes impaires (séparation entre chaque cellule)
     * @param indiceLigne L'indice de chaque ligne de la grille horaire
```

```
     * @param leçon La leçon à traiter
     * @return La représentation sous forme de String de la grille horaire
     */
    private static String créerCellule(int indiceLigne, Leçon leçon) {
        boolean estLignePaire = indiceLigne % 2 == 0;

        //Dans le cas où aucune leçon n'est attribuée, ajouter une cellule vide
        if (leçon == null) {
            if (estLignePaire) {
                return CELLULE_VIDE;
            }
            return BAS_CELLULE;
        }
        //Calculer la période actuelle (de 1 à n)
        int périodeActuelle = indiceLigne / 2 - leçon.périodeDébut + 2;

        //Première période de la leçon : ajouter les informations dans la cellule
        if (périodeActuelle == 1) {
            if (estLignePaire) {
                return String.format(FORMAT_CELLULE, leçon.matière, leçon.salle,
                        leçon.professeur != null ? leçon.professeur.abréviation() : "");
            }
            if (leçon.durée > 1) {
                return CELLULE_VIDE;
            }
        }
        //Périodes suivantes de la leçon : Cellules vides s'il s'agit d'une période intermédiaire
        if (périodeActuelle < leçon.durée || estLignePaire) {
            return CELLULE_VIDE;
        }
        //Cas de la ligne impaire de la dernière période
        return BAS_CELLULE;
    }

    /**
     * Créé l'horaire hebdomadaire en fonction d'une liste de leçons
     * @param leçons Collection de leçons à formater en grille horaire hebdomadaire
     * @return Un String représentant la grille horaire hebdomadaire
     */
    public static String horaire(Collection<Leçon> leçons) {
        //Tableau 2d contenant toutes les leçons
        Leçon[][] tableauLeçons = new Leçon[HEURES.length][Jours.values().length];
        for (Leçon leçon : leçons) {
            for (int i = 0; i < leçon.durée; ++i) {
                tableauLeçons[leçon.périodeDébut - 1 + i][leçon.jourSemaine - 1] = leçon;
            }
        }
        //Construction de la grille horaire
        StringBuilder horaire = new StringBuilder(CréerEnTête());
        for (int i = 0; i < (HEURES.length * 2) - 1; ++i) {
            //Ajout de la première cellule représentant l'heure
            horaire.append(i % 2 != 0 ? SEP_HEURE : String.format(FORMAT_HEURE, HEURES[i / 2]));

            //Ajout du contenu des cellules de chaque heure de début de chaque jour
            for (int j = 0; j < Jours.values().length; ++j) {
                horaire.append(créerCellule(i, tableauLeçons[i / 2][j]));
            }
            horaire.append(SEP_COL + "\n");
        }
        return horaire.append(SEP_LIGNE_COMPLETE).toString();
    }
}
```