Folder src/src

4 printable files

src/src/Groupe.java
src/src/Lecon.java
src/src/Personne.java
src/src/Test.java

src/src/Groupe.java

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Collection;

class Group {
    private int groupNumber;

    private String orientation;

    private int termNumber;

    private List<Student> students;

    private ArrayList<Lesson> lessons;

    /**
     * Create a group
     *
     * @param number      the number of the group
     * @param orientation the orientation of the group
     * @param term        the term of the group
     * @param students    the students of the group
     */
    public Group(int number, String orientation, int term, List<Student> students) {
        this.students = new ArrayList<>();
        this.groupNumber = number;
        this.orientation = orientation;
        this.termNumber = term;
        this.students.addAll(students);
        lessons = new ArrayList<>();
        for (Student student : this.students) {
            student.setGroup(this);
        }
    }

    /**
     * Get the number of students in the group
     *
     * @return the number of students
     */
    public int studentCount() {
        return students.size();
    }

    /**
     * Get the name of the group
     *
     * @return the name of the group
     */
    public String name() {
        return this.orientation + termNumber + "-" + groupNumber;
    }
    /**
     * Define a lesson
     */
    public void defineLesson() {
        System.out.println("Lessons defined!");
    }

    /**
     * Define lessons
     *
     * @param lessons the lessons to define
     */
    public void defineLessons(Collection<Lesson> lessons) {
        this.lessons.addAll(lessons);
    }

    /**
     * Get the schedule of the group
     *
     * @return the schedule of the group
     */
    public String schedule() {
        return "-- Schedule for group " + name() + " (" + studentCount() + " students)\n" +
                Lesson.schedule(lessons);
    }

    /**
     * Get the schedule of the group
     *
     * @return the schedule of the group
     */
    @Override
    public String toString() {
        return "Group " + name() + ": " + studentCount() + " students, schedule: " + schedule();
    }
}
```

src/src/Lecon.java

```java
// Lesson.java

import java.util.Collection;

class Lesson {

    private enum Days {Mon, Tue, Wed, Thu, Fri}

    private static final String[] HOURS = {"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 COL_SEP = '|', LINE_SEP = '-';

    private static final int FIRST_COL_WIDTH = 5,
            SUBJECT_INITIALS = 5, ROOM_INITIALS = 3,
            COL_WIDTH_DAYS = 2 + Teacher.INITIALS_COUNT + SUBJECT_INITIALS + ROOM_INITIALS;

    private static final String CELL_BOTTOM = COL_SEP + (LINE_SEP + "").repeat(COL_WIDTH_DAYS),
            EMPTY_CELL = CELL_BOTTOM.replace(LINE_SEP, ' '),
            TIME_SEP = " ".repeat(FIRST_COL_WIDTH),
            COMPLETE_LINE_SEP = TIME_SEP + CELL_BOTTOM.repeat(Days.values().length) + COL_SEP +
                    "\n",
            TIME_FORMAT = "%" + FIRST_COL_WIDTH + "s",
            CELL_FORMAT = COL_SEP + "%-" + SUBJECT_INITIALS + "s %" + ROOM_INITIALS + "s %" +
                    Teacher.INITIALS_COUNT + "s";
    private final String subject;

    private final int dayOfWeek;

    private final int startPeriod;

    private final int duration;

    private final String room;

    private Teacher teacher;

    /**
     * Create a lesson
     *
     * @param subject     the subject of the lesson
     * @param dayOfWeek   the day of the week of the lesson
     * @param startPeriod the start period of the lesson
     * @param duration    the duration of the lesson
     * @param room        the room of the lesson
     */
    public Lesson(String subject, int dayOfWeek, int startPeriod, int duration, String room) {
        if (startPeriod + duration > HOURS.length) {
            throw new RuntimeException("The duration of the lesson exceeds the schedule length");
        }
        if (duration < 1 || startPeriod < 1) {
            throw new RuntimeException("Duration and start period must be > 0");
        }
        if (dayOfWeek < 1 || dayOfWeek > Days.values().length) {
            throw new RuntimeException("The day of the week must be between 1 and " +
                    Days.values().length);
        }
        if (room.length() > ROOM_INITIALS) {
            throw new RuntimeException("Room initials cannot exceed " +
                    ROOM_INITIALS + " characters. Current: " + room.length());
        }
        if (subject.length() > SUBJECT_INITIALS) {
            throw new RuntimeException("Subject initials cannot exceed " +
                    SUBJECT_INITIALS + " characters. Current: " + subject.length());
        }

        this.subject = subject;
        this.dayOfWeek = dayOfWeek;
        this.startPeriod = startPeriod;
        this.duration = duration;
        this.room = room;
    }

    /**
     * Create a lesson
     *
     * @param subject         the subject of the lesson
     * @param dayOfWeek       the day of the week of the lesson
     * @param startPeriod     the start period of the lesson
     * @param duration        the duration of the lesson
     * @param room            the room of the lesson
     * @param assignedTeacher the teacher assigned to the lesson
     */
    public Lesson(String subject, int dayOfWeek, int startPeriod, int duration, String room, Teacher
assignedTeacher) {
        this(subject, dayOfWeek, startPeriod, duration, room);
        if (assignedTeacher != null) {
            teacher = assignedTeacher;
            teacher.assignLesson(this);
        }
```

```java
}

/**
 * Create the header of the schedule
 *
 * @return the header of the schedule
 */
private static StringBuilder createHeader() {
    StringBuilder header = new StringBuilder(TIME_SEP);
    for (Days day : Days.values()) {
        header.append(String.format(COL_SEP + " %-" + (COL_WIDTH_DAYS - 1) + "s",
                day.name()));
    }
    return header.append(COL_SEP + "\n").append(COMPLETE_LINE_SEP);
}

/**
 * Create a cell of the schedule
 *
 * @param rowIndex the index of the row
 * @param lesson   the lesson to display
 * @return the cell of the schedule
 */
private static String createCell(int rowIndex, Lesson lesson) {
    boolean isEvenRow = rowIndex % 2 == 0;

    if (lesson == null) {
        if (isEvenRow) {
            return EMPTY_CELL;
        }
        return CELL_BOTTOM;
    }
    int currentPeriod = rowIndex / 2 - lesson.startPeriod + 2;

    if (currentPeriod == 1) {
        if (isEvenRow) {
            return String.format(CELL_FORMAT, lesson.subject, lesson.room,
                    lesson.teacher != null ? lesson.teacher.abbreviation() : "");
        }
        if (lesson.duration > 1) {
            return EMPTY_CELL;
        }
    }
    if (currentPeriod < lesson.duration || isEvenRow) {
        return EMPTY_CELL;
    }
    return CELL_BOTTOM;
}

/**
 * Create the schedule
 *
 * @param lessons the lessons to display
 * @return the schedule
 */
public static String schedule(Collection<Lesson> lessons) {
    Lesson[][] lessonGrid = new Lesson[HOURS.length][Days.values().length];
    for (Lesson lesson : lessons) {
        for (int i = 0; i < lesson.duration; ++i) {
            lessonGrid[lesson.startPeriod - 1 + i][lesson.dayOfWeek - 1] = lesson;
        }
    }

    StringBuilder schedule = new StringBuilder(createHeader());
    for (int i = 0; i < (HOURS.length * 2) - 1; ++i) {
        schedule.append(i % 2 != 0 ? TIME_SEP : String.format(TIME_FORMAT, HOURS[i / 2]));
```

```java
            for (int j = 0; j < Days.values().length; ++j) {
                schedule.append(createCell(i, lessonGrid[i / 2][j]));
            }
            schedule.append(COL_SEP + "\n");
        }
        return schedule.append(COMPLETE_LINE_SEP).toString();
    }
}
```

src/src/Personne.java

```java
// Person.java

import java.util.ArrayList;

class Person {

    private final String lastName;

    private final String firstName;

    /**
     * Create a person
     *
     * @param lastName  the last name of the person
     * @param firstName the first name of the person
     */
    public Person(String lastName, String firstName) {
        this.lastName = lastName;
        this.firstName = firstName;
    }

    /**
     * Get the last name of the person
     *
     * @return the last name of the person
     */
    @Override
    public String toString() {
        return firstName + " " + lastName;
    }
}

class Student extends Person {

    private final int id;

    private Group group;

    /**
     * Create a student
     *
     * @param lastName  the last name of the student
     * @param firstName the first name of the student
     * @param id        the ID of the student
     */
    public Student(String lastName, String firstName, int id) {
        super(lastName, firstName);
        if (id < 0) {
            throw new RuntimeException("ID cannot be < 0");
        }
        this.id = id;
    }

    /**
```

```java
     * Set the group of the student
     *
     * @param group the group of the student
     */
    void setGroup(Group group) {
        this.group = group;
    }

    /**
     * Get the group of the student
     *
     * @return the group of the student
     */
    @Override
    public String toString() {
        return "Student " + super.toString() + " (#" + id + ") - " + group.name();
    }
}

class Teacher extends Person {
    private final String abbreviation;

    private final ArrayList<Lesson> lessons = new ArrayList<>();

    static final int INITIALS_COUNT = 3;

    /**
     * Create a teacher
     *
     * @param lastName     the last name of the teacher
     * @param firstName    the first name of the teacher
     * @param abbreviation the abbreviation of the teacher
     */
    public Teacher(String lastName, String firstName, String abbreviation) {
        super(lastName, firstName);
        this.abbreviation = abbreviation;
    }

    /**
     * Assign a lesson to the teacher
     *
     * @param lesson the lesson to assign
     */
    void assignLesson(Lesson lesson) {
        lessons.add(lesson);
    }

    /**
     * Get the abbreviation of the teacher
     *
     * @return the abbreviation of the teacher
     */
    public String abbreviation() {
        return abbreviation;
    }

    /**
     * Get the schedule of the teacher
     *
     * @return the schedule of the teacher
     */
    @Override
    public String toString() {
        return "Teacher " + super.toString() + " (" + abbreviation + ")";
    }
    /**
     * Get the schedule of the teacher
     *
     * @return the schedule of the teacher
     */
    public String schedule() {
        return "-- Schedule " + this + "\n" + Lesson.schedule(lessons);
    }
}
```

src/src/Test.java

```java
import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        // 1. Define teachers Daniel Rossier (DRE) and Pier Donini (PDO).
        Teacher dre = new Teacher("Rossier", "Daniel", "DRE");
        Teacher pdo = new Teacher("Donini", "Pier", "PDO");

        // 2. Define three lessons of the OOP course (PDO), one lesson for SYE (DRE), and one TIC lesson
(independent project).
        Lesson oop1 = new Lesson("OOP", 3, 6, 2, "H02", pdo);
        Lesson oop2 = new Lesson("OOP", 4, 6, 2, "H02", pdo);
        Lesson oop3 = new Lesson("OOP", 4, 8, 2, "H02", pdo);
        Lesson sye1 = new Lesson("SYE", 1, 1, 2, "G01", dre);
        Lesson sye2 = new Lesson("SYE", 4, 3, 2, "A09", dre);
        Lesson tic1 = new Lesson("TIC", 2, 10, 1, "F06");

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

        // 4. Define an IL6-1 group containing the first four students and an SI6-1 group containing the last
two.
        Group il = new Group(1, "IL", 6, Arrays.asList(john, paul, ringo, george));
        Group si = new Group(1, "SI", 6, Arrays.asList(roger, david));

        // 5. Assign all existing lessons to the IL6-1 group. Assign only OOP lessons to the SI6-1 group.
        il.defineLessons(Arrays.asList(oop1, oop2, oop3, sye1, sye2, tic1));
        si.defineLessons(Arrays.asList(oop1, oop2, oop3));

        // 6. Create an array with all persons and display elements using dynamic binding.
        System.out.println("-- Members of the school");
        for (Person p : new Person[]{pdo, dre, john, paul, ringo, george, roger, david}) {
            System.out.println(p);
        }
        System.out.println();

        // 7. Display information about IL6-1 group (name, number of students, schedule).
        System.out.println(il.schedule());

        // 8. Display schedule of teacher PDO.
        System.out.println(pdo.schedule());
    }
}
```