# Institute of Technology of Cambodia

Department of Information and Communication Engineering

# Assignment: TP0

## Subject: **Software Engineering**


Lecturer Course: **TAL Tongsreng**

Lecturer TP: **Roeun Pacharoth**

Student: **Saren Sokmeak**

ID: **e20211376**

Year: **I4**


Academic year: 2024-2025

## TP00.1. Reservation class

Create the Reservation class, which represents the room reservation at department. This class will contain information such as room number, person name who make reservation, date and time start and end of reservation, and other remarks as a string.

```java
class Reservation {
    private String roomNumber; // unique
    private String roomNumberToUpdate;
    private String name;
    private LocalDateTime startDate;
    private LocalDateTime endDate;
    private String otherNote; // new field for additional notes
}
```

# TP00.2. Constructors

```java

    Reservation(String roomNum, String name, String startDate, String endDate, String otherNote) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
       this.roomNumber = roomNum;
       this.name = name;
       this.startDate = LocalDateTime.parse(startDate, formatter);
       this.endDate = LocalDateTime.parse(endDate, formatter);
       this.otherNote = otherNote;
    }

    // delete constructor
    Reservation(String roomNum) {
       this.roomNumber = roomNum;
    }

    // update constructor
    Reservation(String roomNumberToUp, String roomNum, String name, String startDate, String endDate,
            String otherNote) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        this.roomNumberToUpdate = roomNumberToUp;
        this.roomNumber = roomNum;
        this.name = name;
        this.startDate = LocalDateTime.parse(startDate, formatter);
        this.endDate = LocalDateTime.parse(endDate, formatter);
        this.otherNote = otherNote;
    }

```

# TP00.3. Getters/Setters or Accessors/Mutators

```java


public String getRoomNumberToUpdate() {
    return roomNumberToUpdate;
}

public void setRoomNumberToUpdate(String newRoomNumber) {
    if (ValidateRoomNumber(newRoomNumber)) {
        this.roomNumberToUpdate = newRoomNumber;
    } else {
        this.roomNumberToUpdate = "";
    }
}

public LocalDateTime getStartDate() {
    return startDate;
}

public void setStartDate(LocalDateTime startDate) {
    this.startDate = startDate;
}

public void setStartDate(String startDate) {
    if (ValidationTime(startDate)) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        this.startDate = LocalDateTime.parse(startDate, formatter);
    } else {
        this.startDate = null;
    }
}

```

```java

public LocalDateTime getEndDate() {
    return endDate;
}

public void setEndDate(String startDate, String endDate) {
    if (ValidationTimeEnd(startDate, endDate)) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        this.endDate = LocalDateTime.parse(endDate, formatter);
    } else {
        this.endDate = null;
    }
}

public String getRoomNumber() {
    return roomNumber;
}

public void setRoomNumber(String roomNumber) {
    if (ValidateRoomNumber(roomNumber)) {
        this.roomNumber = roomNumber;
    } else {
        this.roomNumber = "";
    }
}

public String getName() {
    return name;
}

public void setName(String name) {
    if (ValidationName(name)) {
        this.name = name;
    } else {
        this.name = "ADMIN";
    }
}

public String getOtherNote() {
    return otherNote;
}

public void setOtherNote(String otherNote) {
    this.otherNote = otherNote; // set otherNote
}
```

## TP00.4. Validations with Exception

1. Room number must not be empty and must start with a letter follows by dash and 3 numbers. Ex: F-209, J-704, …

```java


private Boolean ValidateRoomNumber(String roomNum) {
    String roomPattern = "^[A-Za-z]-\\d{3}$";
    if (roomNum.matches(roomPattern)) {
        // System.out.println("Room number is valid!");
        return true;
    } else {
        // System.out.println("Error, Room number is invalid!");
        return false;
    }
}

```

2. Reservation person's name must not be empty and must contains vowels and consonants.

```java

private Boolean ValidationName(String name) {
    String vowelPattern = ".*[AEIOUaeiou].*";
    String consonantPattern = ".*[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz].*";
    return !name.isEmpty() && name.matches(vowelPattern) && name.matches(consonantPattern);
}

```

3. Reservation date time start must be in the future.

```java

private Boolean ValidationTime(String startTimeString) {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
    try {
        LocalDateTime startTime = LocalDateTime.parse(startTimeString, formatter);
        return startTime.isAfter(LocalDateTime.now());
    } catch (DateTimeParseException e) {
        return false;
    }
}


```

4. Reservation date time end must be greater than date time start at least an hour.

```java


private Boolean ValidationTimeEnd(String startTimeString, String endTimeString) {
    boolean isValidtime = ValidationTime(endTimeString);
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
    if (isValidtime) {
        LocalDateTime startTime = LocalDateTime.parse(startTimeString, formatter);
        LocalDateTime endTime = LocalDateTime.parse(endTimeString, formatter);
        Duration duration = Duration.between(startTime, endTime);
        return duration.toHours() >= 1;
    }
    return false;
}

```

5. Update the constructors to use setters instead of this… = …;

```java

Reservation(String roomNum, String name, String startDate, String endDate, String otherNote) {
    setRoomNumber(roomNum);
    setName(name);
    setStartDate(startDate);
    setEndDate(startDate, endDate);
    setOtherNote(otherNote); // set the otherNote
}

// delete constructor
Reservation(String roomNum) {
    setRoomNumber(roomNum);
}

// update constructor
Reservation(String roomNumberToUp, String roomNum, String name, String startDate, String endDate,
        String otherNote) {
    setRoomNumber(roomNum);
    setRoomNumberToUpdate(roomNumberToUp);
    setName(name);
    setStartDate(startDate);
    setEndDate(startDate, endDate);
    setOtherNote(otherNote); // set the otherNote
}

```

# TP00.5. Va

1. List all reservations.

```java
// Default reservations list:
 ArrayList<Reservation> myReservations = new ArrayList<>();
```

```java
@Override
    public String toString() {
        return "Reservation [roomNumber=" + roomNumber + ", name=" + name + ", startDate=" + startDate + ", endDate="
                + endDate
                + ", otherNote=" + otherNote + "]"; // include otherNote in toString
    }

```

```java
static void ListAllReservations(ArrayList<Reservation> reservations) {
        if (reservations.isEmpty()) {
            System.out.println("No reservations found.");
        } else {
            reservations.forEach(res -> {
                System.out.println(res.toString());
            });
        }
    }
```

2. Add new reservation. Reservation date and time must be expressed as a future time.

```java

System.out.print("Enter Room number: ");
String roomNum = scanner.nextLine();

System.out.print("Enter Customer's name: ");
String name = scanner.nextLine();

System.out.print("Enter Start Date (yyyy-MM-dd HH:mm): ");
String startDate = scanner.nextLine();

System.out.print("Enter End Date (yyyy-MM-dd HH:mm): ");
String endDate = scanner.nextLine();

System.out.print("Enter other remark (optional): ");
String otherRemark = scanner.nextLine();

 // calling constructor for create
Reservation newReservation = new Reservation(roomNum, name, startDate, endDate, otherRemark);
myReservations.add(newReservation);

```

3. Cancel/remove reservation. A Reservation can be canceled/removed if and only if it is not yet started.

```java

public boolean hasStarted() {
    Boolean isStarted = LocalDateTime.now().isAfter(startDate);
    return isStarted;
}

```

```java


public boolean cancelReservation(Reservation res, ArrayList<Reservation> reservationsList) {
    // String roomNumbetoCancel;

    Reservation reservation = null;

    for (Reservation rs : reservationsList) {
        if (rs.getRoomNumber().equals(res.getRoomNumber())) {
            reservation = rs;
            if (!reservation.hasStarted()) {
                reservationsList.remove(reservation);
                return true;
            } else {
                System.out.println("Failed to Cancel cuz the reservatio has started!");
            }
            break;
        }

    }

    return false;
}

```

4. Update reservation if it is not yet started.

```java

public boolean updateReservation(ArrayList<Reservation> arrRes, Reservation resToUpdate) {
    // use the #room to search that reservation
    // String roomNumberToUpdate =reservation.getRoomNumber();

    Reservation res = null;

    boolean isFound = false;

    for (Reservation rs : arrRes) {
        if (rs.getRoomNumber().equals(resToUpdate.getRoomNumberToUpdate())) {
            res = rs;
            isFound = true;

        }
    }
    if (isFound) {
        //
        System.out.println("The old reservation: " + res.toString());

    } else {
        System.out.println("Reservation not found!");
    }

    // DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
    // HH:mm");
    if (!res.hasStarted()) {
        res.setRoomNumber(resToUpdate.getRoomNumber());
        res.setName(resToUpdate.getName());
        res.setStartDate(resToUpdate.getStartDate());
        res.setOtherNote(resToUpdate.getOtherNote()); // update otherNote

        System.out.println("The new reservation: " + res.toString());

        return true;
    }
    return false;
}
```

5. Swap room between 2 reservations of the same date and time reservation

```java
public boolean swapRooms(Reservation res1, Reservation res2) {
    if (res1.getStartDate().equals(res2.getStartDate()) && res1.getEndDate().equals(res2.getEndDate())) {
        String tempRoom = res1.getRoomNumber();
        res1.setRoomNumber(res2.getRoomNumber());
        res2.setRoomNumber(tempRoom);
        return true;
    }
    return false;
}
```

*Thank you teacher <3*