Test Case for Use

Case 6

```
attend.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String subjectName = null;
        String querySubject = "SELECT subject_name FROM subjects
WHERE idsubjects = ?";

        try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/softengin23_
24", "root", "W45@jqr#8CX");
             PreparedStatement preparedStatementSubject =
connection.prepareStatement(querySubject)) {

            preparedStatementSubject.setString(1, id); // Assuming id
is the subject ID

            try (ResultSet resultSet =
preparedStatementSubject.executeQuery()) {

                if (resultSet.next()) {
                    subjectName = resultSet.getString("subject_name");

                } else {
                    JOptionPane.showMessageDialog(null, "No subject
found with the given ID.");
                    return; // Exit the action listener if no subject
is found
                }
            }

        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Database error: " +
ex.getMessage());
            return; // Exit the action listener if there is a
database error
        }

        String queryMeeting = "SELECT every1, startTime, status FROM
meetings_schedule_subjects WHERE meetingsAbout = ?";

        try (Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/softengin23_
24", "root", "W45@jqr#8CX");
             PreparedStatement preparedStatementMeeting =
connection.prepareStatement(queryMeeting)) {
```

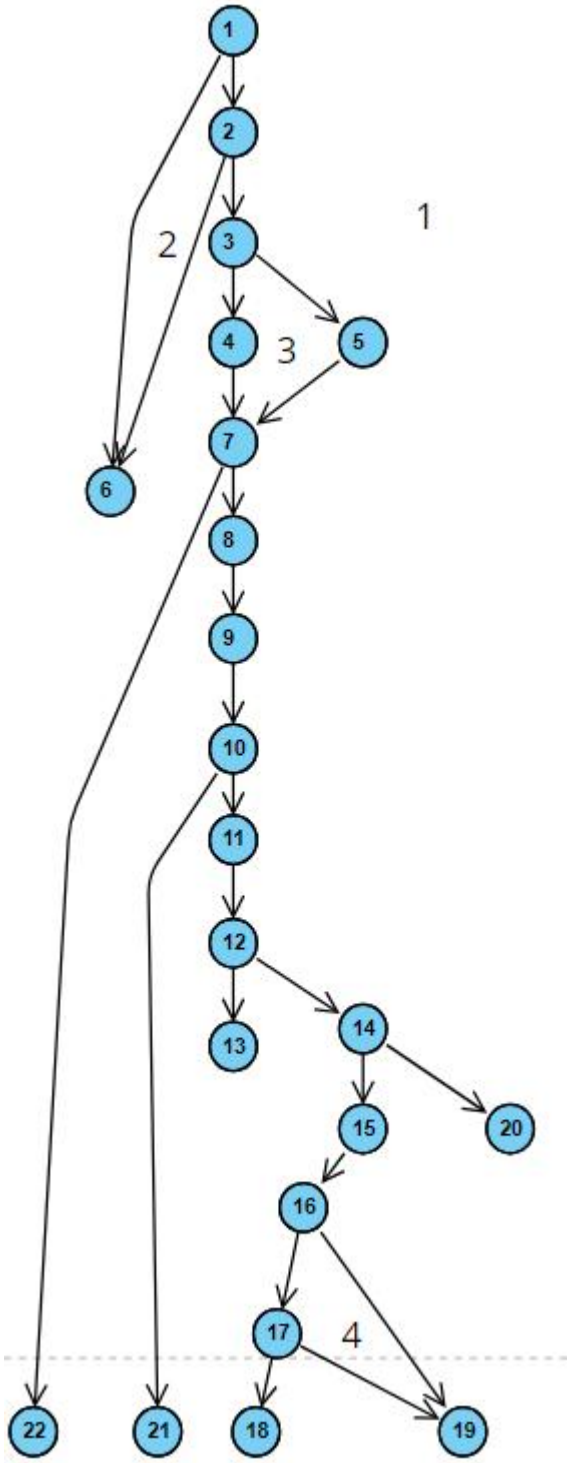
```
            preparedStatementMeeting.setString(1, subjectName);

            try (ResultSet resultSet =
preparedStatementMeeting.executeQuery()) {

                if (resultSet.next()) {
                    String every1 = resultSet.getString("every1");
                    String startTimeStr =
resultSet.getString("startTime");
                    String status = resultSet.getString("status");

                    // Check the status

                    if ("canceled".equalsIgnoreCase(status)) {
                        JOptionPane.showMessageDialog(null, "The
lecture has been canceled. Let me redirect you to studentMenu");
                        Timer timer = new Timer(1000, r -> {
                            dispose();
                            new studentMenuu(null, user);
                        });
                        timer.setRepeats(false);
                        timer.start();

                    } else if ("scheduled".equalsIgnoreCase(status))
{
                        // Get the current day name and time
                        String todayDayName =
DayOfWeek.from(LocalDate.now()).name().toLowerCase();
                        LocalTime startTime =
LocalTime.parse(startTimeStr, DateTimeFormatter.ofPattern("h:mm a",
Locale.ENGLISH));
                        LocalTime currentTime = LocalTime.now();

                        // Calculate the difference in minutes
                        long minutesDifference =
java.time.Duration.between(currentTime, startTime).toMinutes();

                        if
(every1.toLowerCase().contains(todayDayName) && minutesDifference >
30) {
                            JOptionPane.showMessageDialog(null,
"The lecture will begin in " + minutesDifference + " minutes.");
                        } else {
```

8 9 10 11 12 13 14 15 16 17 18

```
                        dispose();
                        new LobbyGUIStudent(null, user, id);
                    }
                } else {
                    dispose();
                    new LobbyGUIStudent(null, user, id);
                }
            } else {
                JOptionPane.showMessageDialog(null, "No meeting schedule found.");
            }
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Database error: " + ex.getMessage());
    }
}
});
```

Για τις συναρτήσεις addUserToLobby, removeUserFromLobby και acceptAllStudents που καλούνται με την είσοδο του Φοιτητή στο Λόμπι και όταν ο Καθηγητής δέχεται τον Φοιτητή αντίστοιχα, δεν χρειάστηκε να δημιουργηθεί κάποιος γράφος, καθώς η ροή του κώδικά τους είναι ενιαία και συνεχής.

Στην επόμενη σελίδα παρουσιάζεται ο Γράφος για το actionListener..



Σύμφωνα με τον τύπο: V(g) = e-n+2 =

= 24-22+2

= 4 περιοχές, όπως φαίνεται και στον γράφο.

Εφαρμόζουμε τον αλγόριθμο για εύρεση μονοπατιών.

Ξεκινάμε με το συντομότερο έγκυρο μονοπάτι:

M1: 1-6

Ακολουθούν τα υπόλοιπα μονοπάτια που προκύπτουν με προσθήκη ακμών στο M1.

M2: 1-2-3-4-7-8-9-10-11-12-13 (Alternative Flow 2)

M3: 1-2-3-4-7-8-9-10-11-12-14-15-16-19 (steps of Main Flow)

M4: 1-2-3-4-7-8-9-10-11-12-14-15-16-17-18 (Alternative Flow 1)

Path	Περιγραφή	Περίπτωση Ελέγχου	Αναμενόμενο Αποτέλεσμα (Έξοδος Προγράμματος)
M2	Έλεγχος για σύνδεση στη Βάση, ανάκτηση δεδομένων για την Κλήση, η κλήση έχει	Connection conn, resultSet.next(), ("canceled".equalsIgnoreCase(status)) == true	JOptionPane.*showMessageDialog*(null, "The lecture has been canceled. Let me redirect you to studentMenu");

	α κ υ ρ ω θ εί		
M 3	Έλ ε γ χ ος για σύν δ ε ση στη Βάση, ανάκ τη ση δεδο μέν ων για την Κλήση, ο χρήσ τη ς εισέρ χε τα ι στο Λόμ πι	Connection conn, resultSet.next(), ("scheduled".equalsIgnoreCase(status) == true, every1.toLowerCase().contains(todayDayName == false	new LobbyGUIStudent(null, user, id);
M4	Έλ ε γ χ ος για ύπ α ρ ξ η γρ α μ μής κά μ ε ρ α ς η ο π οί α	Connection conn, resultSet.next(), ("canceled".equalsIgnoreCase(status) == false, ("scheduled".equalsIgnoreCase(status)) == true every1.toLowerCase().contains(todayDayName) && minutesDifference > 30 == true	JOptionPane.*showMessageDialog*(null, "The lecture will begin in " + minutesDifference + " minutes.");

	ε ί ν α ι κ λ ε ι σ τ ή		