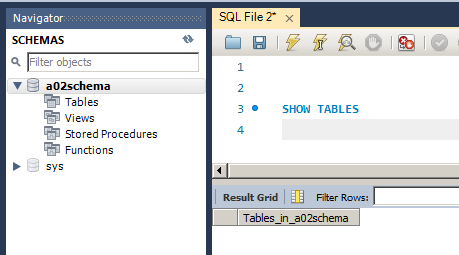
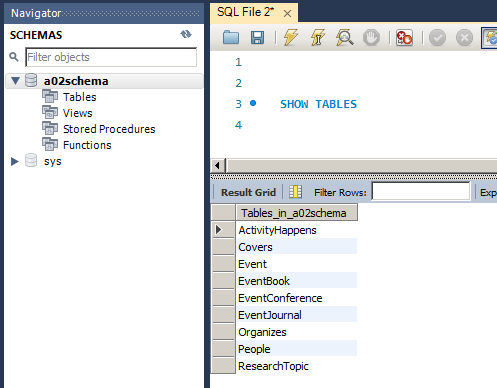
**Question 1**

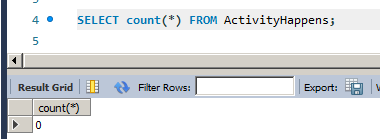
Show the list of tables (there should be none) before the above scripts are executed:

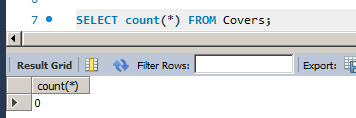


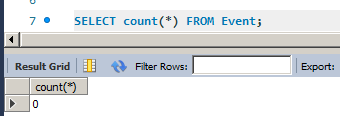
Show the list of tables after run the a02SchemaScript.sql script:

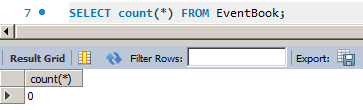


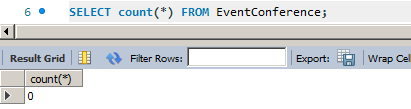
Show the number of rows in each table:

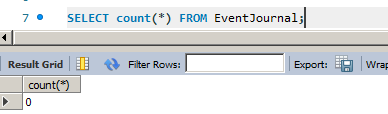


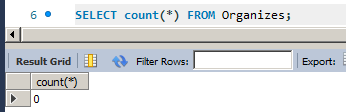


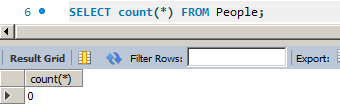


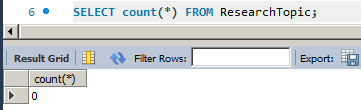




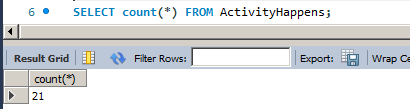


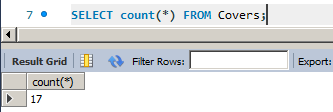


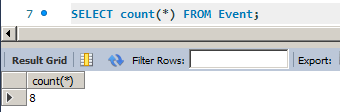


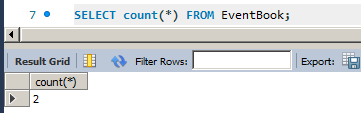


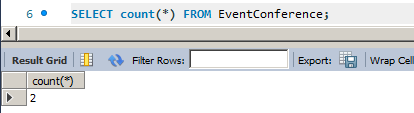
Show the number of rows in each table after executing the script a02DataScript.sql:

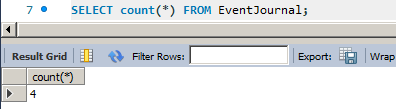


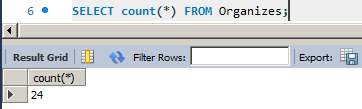


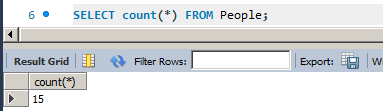


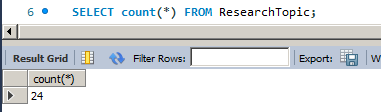












**Questions 2-4**

**A02Main.java**

public class A02Main {

//The purpose of this class is to start the program and display the GUI

public static void main(String[] args) {

A02FrontEnd window = new A02FrontEnd();

window.setVisible(true);

}

}

**A02FrontEnd.java**

import java.awt.GridBagConstraints;

import java.awt.GridBagLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JRadioButton;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

public class A02FrontEnd extends JFrame implements ItemListener, ActionListener{

/\*\*

\* The purpose of this class is to create the FrontEnd/GUI and provide listener functions.

\* You cannot write/perform any database interaction functions/actions in this class.

\* You can only invoke a suitable function from A02MiddleTier class on the click event of Submit button.

\*/

JCheckBox eventConference;

JCheckBox eventJournal;

JCheckBox eventBook;

JRadioButton allDates;

JRadioButton dateRange;

ButtonGroup dateSelection;

JTextArea queryOutput;

JLabel fromLabel;

JLabel toLabel;

JTextField fromDate;

JTextField toDate;

JButton submitQuery;

A02MiddleTier a02MiddleTier;

public A02FrontEnd() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

this.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

a02MiddleTier = new A02MiddleTier();

eventConference = new JCheckBox("EventConference");

eventConference.setSelected(false);

eventJournal = new JCheckBox("EventJournal");

eventJournal.setSelected(false);

eventBook = new JCheckBox("EventBook");

eventBook.setSelected(false);

eventConference.addItemListener(this);

eventJournal.addItemListener(this);

eventBook.addItemListener(this);

JPanel eventPanel = new JPanel(new GridLayout(0, 1));

eventPanel.add(eventConference);

eventPanel.add(eventJournal);

eventPanel.add(eventBook);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 0;

c.gridy = 0;

this.add(eventPanel, c);

JLabel emptyLabel1 = new JLabel(" \n ");

JPanel emptyPanel1 = new JPanel(new GridLayout(0, 1));

emptyPanel1.add(emptyLabel1);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 1;

c.gridy = 0;

this.add(emptyPanel1, c);

allDates = new JRadioButton("All Events");

allDates.setSelected(true);

dateRange = new JRadioButton("Period");

dateRange.setSelected(false);

dateSelection = new ButtonGroup();

allDates.addItemListener(this);

dateRange.addItemListener(this);

dateSelection.add(allDates);

dateSelection.add(dateRange);

JPanel datePanel = new JPanel(new GridLayout(0, 1));

datePanel.add(allDates);

datePanel.add(dateRange);

fromLabel = new JLabel(" From ");

fromLabel.setEnabled(false);

toLabel = new JLabel(" To ");

toLabel.setEnabled(false);

fromDate = new JTextField();

fromDate.setEnabled(false);

toDate = new JTextField();

toDate.setEnabled(false);

JPanel dateRangePanel = new JPanel(new GridLayout(0, 4));

dateRangePanel.add(fromLabel);

dateRangePanel.add(fromDate);

dateRangePanel.add(toLabel);

dateRangePanel.add(toDate);

datePanel.add(dateRangePanel);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 2;

c.gridy = 0;

this.add(datePanel, c);

JLabel emptyLabel2 = new JLabel(" \n ");

JPanel emptyPanel2 = new JPanel(new GridLayout(0, 1));

emptyPanel2.add(emptyLabel2);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 1;

c.gridy = 1;

this.add(emptyPanel2, c);

submitQuery = new JButton("Submit");

submitQuery.addActionListener(this);

JPanel submitPanel = new JPanel(new GridLayout(0, 1));

submitPanel.add(submitQuery);

c.fill = GridBagConstraints.NONE;

c.anchor = GridBagConstraints.LINE\_START;

c.gridx = 2;

c.gridy = 2;

this.add(submitPanel, c);

JLabel emptyLabel3 = new JLabel(" \n ");

JPanel emptyPanel3 = new JPanel(new GridLayout(0, 1));

emptyPanel3.add(emptyLabel3);

c.fill = GridBagConstraints.HORIZONTAL;

c.gridx = 1;

c.gridy = 3;

this.add(emptyPanel3, c);

queryOutput = new JTextArea(18,50);

queryOutput.setText("The\nQuery\nOutput\nwill\nAppear\nhere.");

queryOutput.setText("");

JScrollPane outputPanel = new JScrollPane(queryOutput);

c.gridwidth = GridBagConstraints.REMAINDER;

c.fill = GridBagConstraints.BOTH;

c.gridx = 0;

c.gridy = 4;

this.add(outputPanel, c);

this.setBounds(50, 50, 600, 500);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

//this function is the listener for three check boxes and two radio buttons

public void itemStateChanged(ItemEvent e) {

Object source = e.getItemSelectable();

if (source == eventConference) {

System.out.println("1");

} else if (source == eventJournal) {

System.out.println("2");

} else if (source == eventBook) {

System.out.println("3");

} else if (source == dateRange) {

if (dateRange.isSelected()) {

fromLabel.setEnabled(true);

toLabel.setEnabled(true);

fromDate.setEnabled(true);

toDate.setEnabled(true);

}

else {

fromLabel.setEnabled(false);

toLabel.setEnabled(false);

fromDate.setEnabled(false);

toDate.setEnabled(false);

}

System.out.println("4");

} else if (source == allDates) {

System.out.println("5");

}

}

private void displayResult(java.util.List<String> result){

if(queryOutput.getText().trim().length() > 0){

queryOutput.setText(queryOutput.getText()+"\n" + "Query:\n" + result.get(0));

} else {

queryOutput.setText("Query:\n" + result.get(0));

}

queryOutput.setText(queryOutput.getText()+"\n" + "Output:");

if(result.size() <= 2){

queryOutput.setText(queryOutput.getText()+"\n" + "Empty set");

} else {

for(int i = 1;i < result.size();i++){

queryOutput.setText(queryOutput.getText()+"\n" + result.get(i));

}

}

queryOutput.setText(queryOutput.getText()+"\n");

}

/\*\* Listens to the submit button click \*/

public void actionPerformed(ActionEvent e) {

try {

String fromDateString = fromDate.getText().trim();

String toDateString = toDate.getText().trim();

if(eventConference.isSelected() && allDates.isSelected()){

displayResult(a02MiddleTier.getAllEventConference());

}

if(eventJournal.isSelected() && allDates.isSelected()){

displayResult(a02MiddleTier.getAllEventJournal());

}

if(eventBook.isSelected() && allDates.isSelected()){

displayResult(a02MiddleTier.getAllEventBook());

}

if(allDates.isSelected() && !eventBook.isSelected() && !eventJournal.isSelected() && !eventConference.isSelected()){

displayResult(a02MiddleTier.getAllEvent());

}

// query by date range

if(eventConference.isSelected() && dateRange.isSelected()){

displayResult(a02MiddleTier.getAllEventConferenceByDateRange(fromDateString, toDateString));

}

if(eventJournal.isSelected() && dateRange.isSelected()){

displayResult(a02MiddleTier.getAllEventJournalByDateRange(fromDateString, toDateString));

}

if(eventBook.isSelected() && dateRange.isSelected()){

displayResult(a02MiddleTier.getAllEventBookByDateRange(fromDateString, toDateString));

}

if(dateRange.isSelected() && !eventBook.isSelected() && !eventJournal.isSelected() && !eventConference.isSelected()){

displayResult(a02MiddleTier.getAllEventByDateRange(fromDateString, toDateString));

}

} catch (Exception e1) {

e1.printStackTrace();

}

}

}

**A02MiddleTier.java**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

public class A02MiddleTier {

// This class will contain your code for interacting with Database, acquire the query result and display it in the GUI text area.

// establish connection with the database

public Connection getConnection() throws Exception {

Class.forName("com.mysql.cj.jdbc.Driver");

String URL = "jdbc:mysql://127.0.0.1:3306/a02schema";

Connection connection = DriverManager.getConnection(URL, "root", "root1234");

return connection;

}

public List<String> getAllEvent() throws Exception{

String sql = "select \* from Event";

return query(sql, null, null);

}

public List<String> getAllEventConference() throws Exception{

String sql = "select \* from Event join EventConference on Event.ID = EventConference.EventID";

return query(sql, null, null);

}

public List<String> getAllEventJournal() throws Exception{

String sql = "select \* from Event join EventJournal on Event.ID = EventJournal.EventID";

return query(sql, null, null);

}

public List<String> getAllEventBook() throws Exception{

String sql = "select \* from Event join EventBook on Event.ID = EventBook.EventID";

return query(sql, null, null);

}

public List<String> getAllEventByDateRange(String fromDate, String toDate) throws Exception{

String sql = "select \* from ActivityHappens \n"

+ " join Event on ActivityHappens.EventID = Event.ID\n"

+ "where ActivityDate between ? and ?";

return query(sql, fromDate, toDate);

}

public List<String> getAllEventConferenceByDateRange(String fromDate, String toDate) throws Exception{

String sql = "select \* from ActivityHappens \n"

+ " join Event on ActivityHappens.EventID = Event.ID\n"

+ " join EventConference on Event.ID = EventConference.EventID\n"

+ "where ActivityDate between ? and ?";

return query(sql, fromDate, toDate);

}

public List<String> getAllEventJournalByDateRange(String fromDate, String toDate) throws Exception{

String sql = "select \* from ActivityHappens \n"

+ " join Event on ActivityHappens.EventID = Event.ID\n"

+ " join EventJournal on Event.ID = EventJournal.EventID\n"

+ "where ActivityDate between ? and ?";

return query(sql, fromDate, toDate);

}

public List<String> getAllEventBookByDateRange(String fromDate, String toDate) throws Exception{

String sql = "select \* from ActivityHappens \n"

+ " join Event on ActivityHappens.EventID = Event.ID\n"

+ " join EventBook on Event.ID = EventBook.EventID\n"

+ "where ActivityDate between ? and ?";

return query(sql, fromDate, toDate);

}

// execute sql and return the result, first element in the list is the sql

public List<String> query(String sql, String fromDate, String toDate) throws Exception{

List<String> result = new ArrayList<String>();

Connection connection = getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(sql);

if(fromDate != null){

preparedStatement.setString(1, fromDate);

}

if(toDate != null){

preparedStatement.setString(2, toDate);

}

ResultSet rs = preparedStatement.executeQuery();

result.add(rs.getStatement().toString().split(":")[1].trim());

String title = "";

for(int i = 1;i <= rs.getMetaData().getColumnCount();i++){

if(i > 1){

title += " # ";

}

title += rs.getMetaData().getColumnName(i);

}

result.add(title);

while (rs.next()) {

String row = "";

for(int i = 1;i <= rs.getMetaData().getColumnCount();i++){

if(i > 1){

row += " # ";

}

row += rs.getObject(i).toString().replace('\n', ' ');

}

result.add(row);

}

preparedStatement.close();

connection.close();

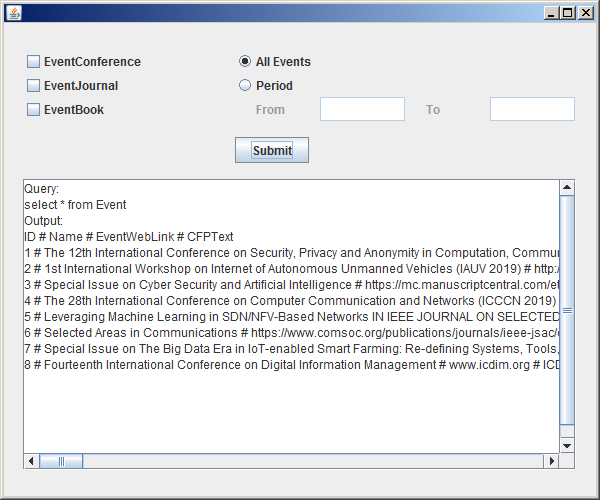
return result;

}

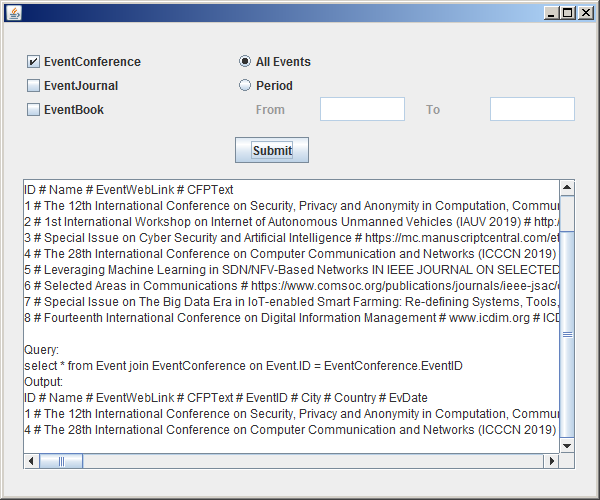
}

**Question 5**

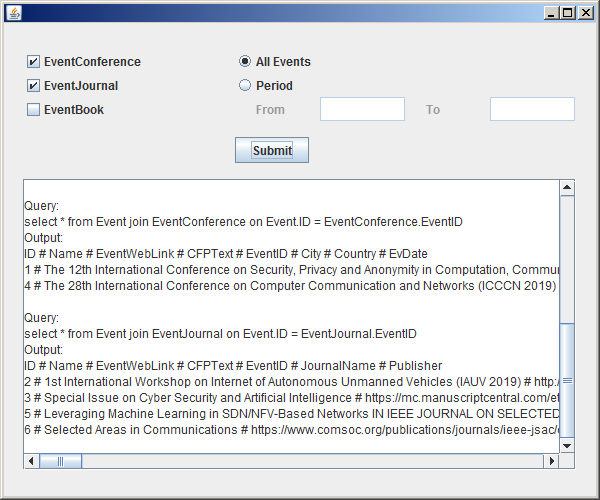
test case 1, only select "All Events":



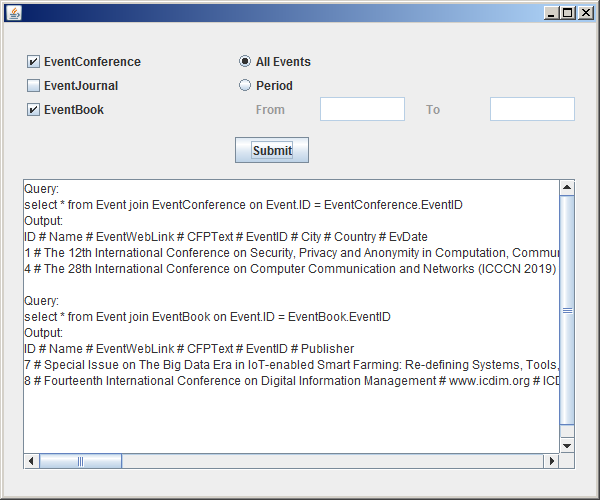
test case 2, select "All Events", "EventConference":



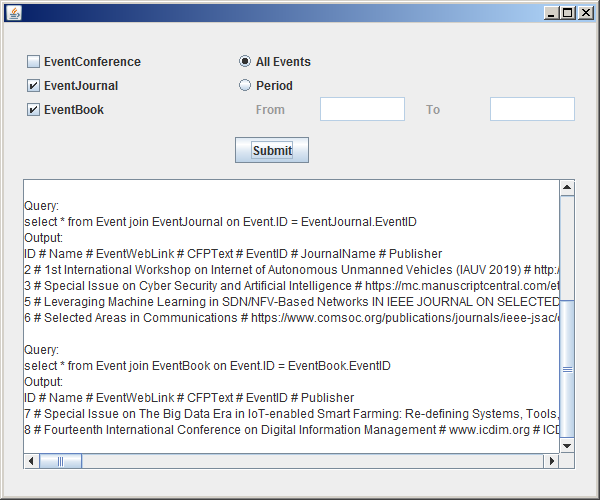
test case 3, select "All Events", "EventConference", "EventJournal":



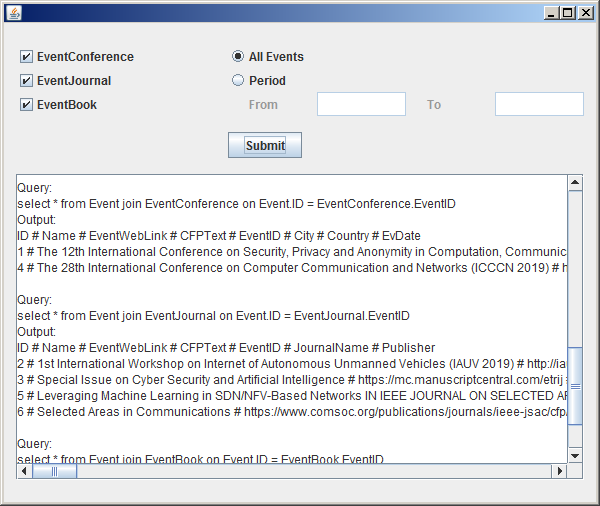
test case 4, select "All Events", "EventConference", "EventBook":

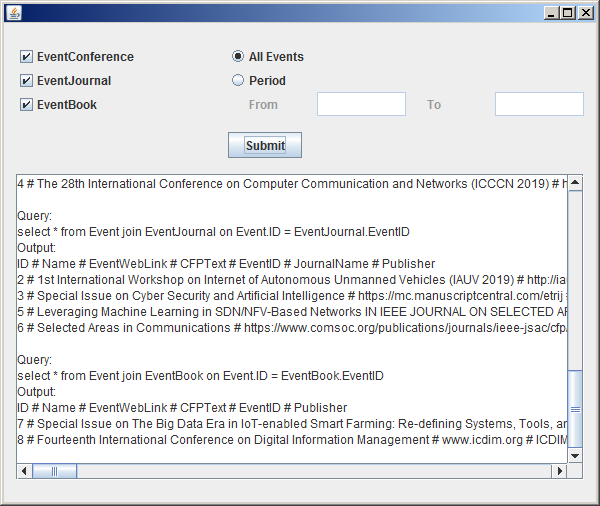


test case 5, select "EventConference", "EventJournal", "EventBook":

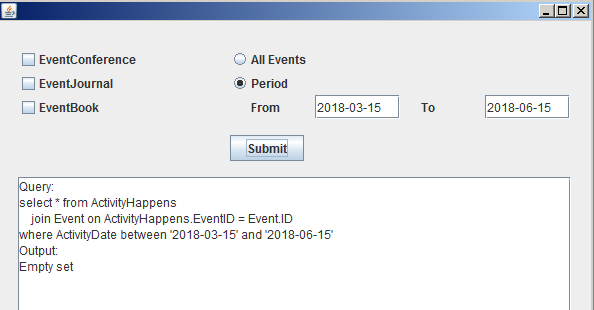


test case 6, select "All Events", "EventConference", "EventJournal", "EventBook":

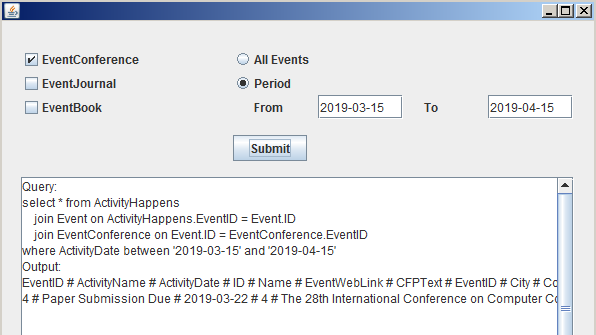




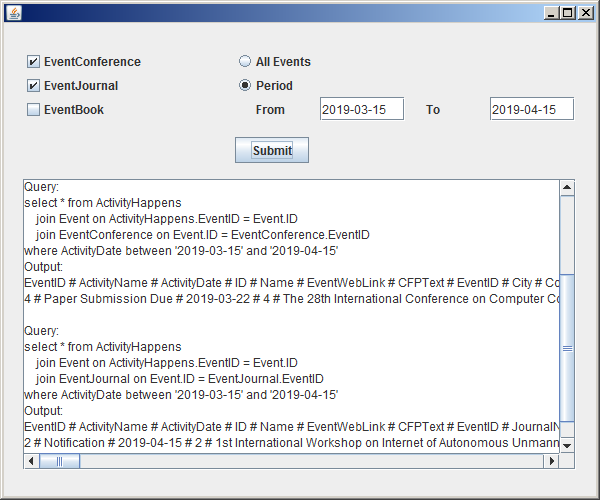
test case 7, select "Period" with date out of range:



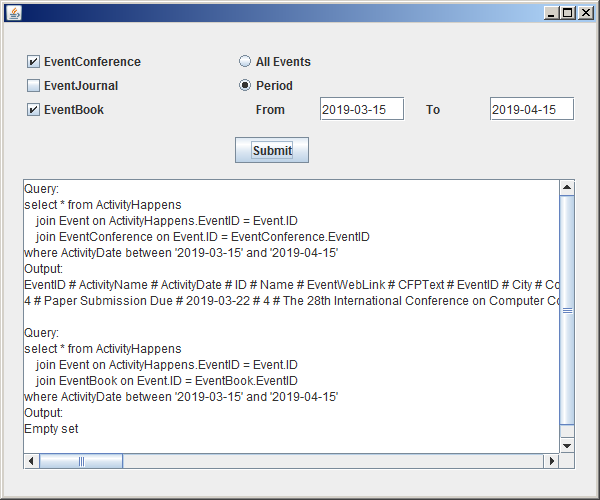
test case 8, select "Period", "EventConference" with valid date range:



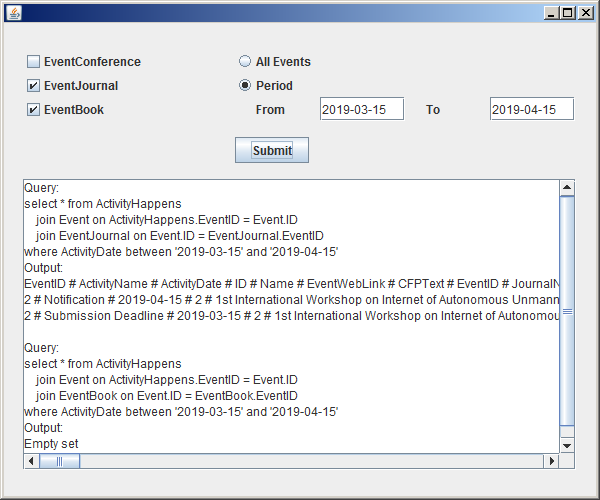
test case 9, select "Period", "EventConference", "EventJournal" with valid date range:



test case 10, select "Period", "EventConference", "EventBook" with valid date range:



test case 11, select "Period", "EventJournal", "EventBook" with valid date range:



test case 12, select "Period", "EventConference", "EventJournal", "EventBook" with valid date range:

