# Lab1: Java Development Environment

In this first lab, we will

- install Eclipse Java IDE (Integrated Development Environment);
- add WindowBuilder to create a window application in 3 steps; and
- add SQLite JDBC drive to create a database application in 3 steps.

Through the Lab1 exercises, you will learn

- how to program in Java;
- how to create a window application with WindowBuilder;
- how to create a database application with SQLite; and
- how to create a multithread application.

## Eclipse Java IDE

1. Go to https://www.eclipse.org/downloads.
2. Download an appropriate version of "Eclipse IDE 2023-03" (or the latest version).
3. Install Eclipse IDE by running the downloaded installer.

## Hello World

1. Run Eclipse IDE.

2. Choose an appropriate workspace directory and launch.

3. Kill the Welcome window by clicking X.

4. File -> New -> Java Project.

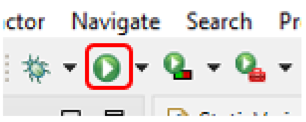
5. Set Project Name to "HelloWorld", uncheck "Create module-info.java file", and click on Finish.

6. File -> New -> Class, set Name to "HelloWorld", and click on Finish.

7. Add the following main method in the HelloWorld class definition.

```
public static void main(String[] args) {
    System.out.println("Hello, world!");
}
```

8. Execute the program by clicking the white triangle in the green circle.



9. Click on OK in the "Save and Launch" window.

10. Check to see that "Hello, world!" is printed in Console.

Note 1: System.out.println() prints a string in one line in Console.

Note 2: Each Java program has to have one main method defined by `public static void main(String[] args) { ... }` in a class. This is the main method to be executed first when the Java program gets executed.

Note 3: No specific requirements on the name of the class with the main method.

## Window Application Step 1

1. Create a new Java project "WinApp1".
2. Help -> Eclipse Market Place.
3. Find "Window Builder".
4. Install "WindowBuilder Current" and click on Confirm.
5. Check "I accept the terms of license agreements" and click on Finish.
6. Wait for installing software; the progress is shown in the bottom right corner.
7. Click on "Restart Now".
8. Wait for the restart of Eclipse IDE.

---

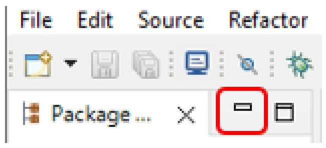
9. Right-click on "WinApp1" -> New -> Other.
10. WindowBuilder -> Swing Designer -> Application Window and click on Next.
11. Set Name to "WinApp1" and click on Finish.

---

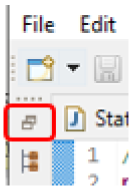
12. Click on Design below the code pane.

13. Adjust the panes to clearly see the blank window.

14. Minimize Package Explorer on the left side by clicking the white rectangle next to the Package Explorer.



15. Get Package Explorer back by clicking the following symbol on the left side.



16. Righ-click on the blank window -> Set layout -> Absolute layout.

17. Click on JLabel in Parette/Components, move the cursor to the blank window, and enlarge the JLabel box.

18. Press ESC to remove the JLabel message box if necessary.

19. Change Variable name to "lblClock" in Properties.

20. Change the font size to 14 in Properties.

21. Back to Source by clicking on Source below the Designer pane.

22. `lblClock` is declared locally at the end of `initialize()`.

23. Add the declaration `private JLabel lblClock;` after `private JFrame frame;` at the top of the `WinApp11` class definition.

```
private JFrame frame;
private JLabel lblClock;
```

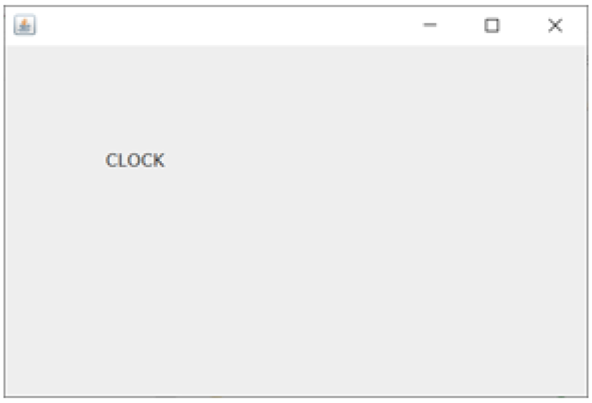
24. Remove `JLabel` from `JLabel lblClock= new JLabel("New label");` at the end of `initialize()`.

```
lblClock = new JLabel("New Label");
lblClock.setFont(new Font("Tahoma", Font.PLAIN, 14));
```

25. After `initialize();` in the class constructor `WinApp1()`, add the following method call to change the label name.

```
initialize();
lblClock.setText("CLOCK");
```

26. Run the program by clicking on the white triangle in the green circle.

27. Check to see that "CLOCK" appears in the popped-up window.


28. Close the window by clicking on X at the top-right corner of the window.

Note 1: When you reopen the source file created by Application Window, you may not see the Designer tab under the source window. In that case, right-click on the file and choose Open with -> WindowBuilder editor.

Note 2: If the source file created by "Application Window" is opened with the regular Java editor, you don't see the Designer tab.

## Window Application Step 2

1. Create a new Java project "WinApp2".
2. Copy WinApp1/src/WinApp1.java and past it WinApp2/src.
3. Right-click on WinApp2/src/WinApp1.java.
4. Refactor -> Rename, chane name to WinApp2, click on Finish, and click on Finish again.
5. Double-click on WinApp2/src/WinApp2.java.
6. Run the program to see "CLOCK" in the popped-up window and close it.

---

7. Go to Designer.

8. Add the following two import statements after the last import statement at the beginning of the source file.

```
 import java.util.Calendar;
 import java.util.GregorianCalender;
```

9. In the class constructor `WinApp2()`, comment out `lblClock.setText("CLOCK");` by adding `//`.

10. Add "clock();" after that.

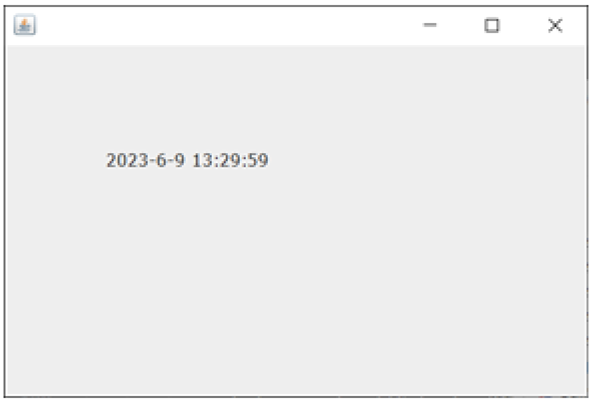
```
initialize();
// lblClock.setText("CLOCK");
clock();
```

11. After the constructor definition `WindApp2()`, add the following method.

```
public void clock() {
    Calendar cal = new GregorianCalendar();
    int year = cal.get(Calendar.YEAR);
    int month = cal.get(Calendar.MONTH) + 1;
    int day = cal.get(Calendar.DAY_OF_MONTH);
    int hour = cal.get(Calendar.HOUR_OF_DAY);
    int minute = cal.get(Calendar.MINUTE);
    int second = cal.get(Calendar.SECOND);
    lblClock.setText(year + "-" + month + "-" + day + " " + hour + ":" +
minute + ":" + second);
}
```

12. Run the program.

13. Check to see that the current time is correctly printed in the popped-up window. It should look like:


2023-6-9 13:29:59

14. Enlarge the clock (JLabel) box if necessary.

## Window Application Step 3

1. Create a new project "WinApp3".
2. Copy WinApp2/src/WinApp2.java to WinApp3/src.
3. Rename the file to WinApp3 using Refactor -> Rename.

---

4. Go to Designer.

5. Add JButton from Parette/Components to the window.

6. Change Variable to "btnClock". text to "Clock On". and font size to 14.

7. Go back to Source.

8. Add the following statement at the beginning of the WinApp class definition.

```
private boolean clockOn;
```

9. After `WinApp3()`, add the following `ClockThread` class definition.

```
public class ClockThread extends Thread {
   public void run() {
       while (true) {
           clock();
           try {
               sleep(1000); // 1000 ms
               if (!clockOn) break;
           }
           catch (Exception e) {
               System.out.println("<<< " + e.getMessage());
               break;
           }
       }
   }
}
```

10. Go to Designer.

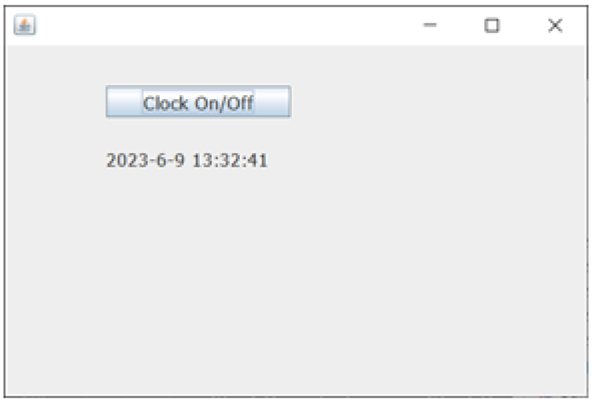
11. Double-click on btnClock.

12. Add the following statements in the body of `actionPerformed(ActionEvent e)`.

```
if (clockOn) {
    btnClock.setText("Clock On);
    clockOn = false;
}
else {
    btnClock.setText("Clock Off");
    clockOn = true;
    ClockThread ct = new ClockThread();
    ct.start();
}
```

13. Run the program.

14. Check that the click on "Clock On" turns the clock on and the click on "Clock Off" turns the clock off.



Note 1: The `ClockThread` class is a thread class that inherits from `Thread`.

Note 2: The `run()` method of the `ClockThread` class is executed when the `start()` of the class instance is called.

## Database Application Step 1

1. Create a new Java project "DbApp1".
2. File -> New -> Other. WindowBuilder -> Swing Designer -> Application Window, click on Next, set name to DbApp1.
3. Go to Designer.
4. Click on the blank window, Set Layout -> Absolute layout.
5. Create a JLabel; set Variable to "lblDbName", text to "SQLite DB Name", and font size to 14 (or an appropriate font size).
6. Create a JTextField; set Variable to "txtDbName", and text to the name of the database to be created; in this example we use `sqlite1.db`.

7. Create a JLabel; set Variable to "lblCsvFile" and text to "CSV Data File".
8. Create a JTextField; set Variable to "txtCsvFile" and text to the prepared CSV file of tennis players; it is stored in `../Lab1_Files/tennis_players.csv` for this example.
9. Create a JButton; set Variable to "btnInit" and text to "Init".
10. Create a JButton; set Variable to "btnLoad" and text to "Load".
11. Create a JComboBox; set Variable to "comboBoxList".

---

12. Go to https://github.com/xerial/sqlite-jdbc.
13. Click on Release 3.41.2.1 (or a newer version).
14. Download the JAR file (sqlite-jdbc-3.41.2.1.jar or a newer version).
15. Right-click on DbApp1 -> JRE System Library.
16. Build Path -> Configure Build Path...
17. Scoll up and click on Modulepath.
18. Click on Add External JARs.
19. Choose the downloaded sqlite-jdbc jar file.
20. Click on Apply and Close.

---

21. File -> New -> Class; set Name to "DbData".

22. Define the `DbData` class as follows.

```
import java.io.File;
import java.util.Scanner;
import java.util.ArrayList;

public class DbData {
    public static ArrayList<String[]> readFile(String fname) {
        ArrayList<String[]> data = new ArrayList<String[]>();
        try {
            File fobj = new File(fname);
            Scanner scan = new Scanner(fobj);
            while (scan.hasNextLine()) {
                String line = scan.nextLine();
                String[] itemList = line.split("[,]", 0);
                if (itemList.length != 8) continue;
                for (int i = 0; i < itemList.length; i++) itemList[i] =
itemList[i].trim();
                data.add(itemList);
            }
            scan.close();
        }
        catch (Exception e) {
            System.out.println("<<< " + e.getMessage());
        }
        return data;
    }
}
```

23. The `readFile` method defined in this class reads the initial database data from a CSV file and returns the data in `ArrayList`.

---

24. File -> New -> Class; set Name to "SqliteDb".

25. Define the `SqliteDb` class as follows.

```
import java.sql.*;
import javax.swing.*;
import java.util.ArrayList;

public class SqliteDb {

    private static final String URL = "jdbc:sqlite:";
    private static String dbUrl = null;
    // TennisPlayer Field Index
    public static final int TPF_TPID         = 0;
    public static final int TPF_LASTNAME     = 1;
    public static final int TPF_FIRSTNAME    = 2;
    public static final int TPF_AGE          = 3;
    public static final int TPF_COUNTRY      = 4;
    public static final int TPF_ORGANIZATION = 5;
    public static final int TPF_RANKING      = 6;
    public static final int TPF_POINTS       = 7;
    public static final int TPF_COUNT        = 8;
    public static String[] TPF_NAME = {
        "Tpid",
        "LastName",
        "FirstName",
        "Age",
        "Country",
        "Organization",
        "Ranking",
        "Points"
    };

    public static void setDb(String dbname) {
        dbUrl = URL + dbname;
    }

    public static Connection connect() {
        Connection conn = null;
        try {
            Class.forName("org.sqlite.JDBC");
            conn = DriverManager.getConnection(dbUrl);
        }
        catch (Exception e) {
            System.out.println("<<< " + e.getMessage());
            JOptionPane.showMessageDialog(null, "<<< Connection to SQLite DB
<" + dbUrl + "> Failed");
        }
        return conn;
    }

    public static void createTable() {
        Connection conn = connect();
        if (conn == null) return;
        String sql1 = "CREATE TABLE IF NOT EXISTS TennisPlayers (\n" +
            "Tpid         TEXT PRIMARY KEY,\n" +
            "LastName     TEXT NOT NULL,\n" +
            "FirstName    TEXT NOT NULL,\n" +
            "Age          TEXT NOT NULL,\n" +
            "Country      TEXT NOT NULL,\n" +
            "Organization TEXT NOT NULL,\n" +
            "Ranking      TEXT NOT NULL,\n" +
            "Points       TEXT NOT NULL\n" +
        ");";
        try {
            Statement st = conn.createStatement();
            st.execute(sql1);
            st.close();
            conn.close();
        }
        catch (Exception e) {
            System.out.println("<<< " + e.getMessage());
            JOptionPane.showMessageDialog(null, "<<< Table Creation in
SQLite DB <" + dbUrl + "> Failed");
            return;
        }
        JOptionPane.showMessageDialog(null, "Created a new table in SQLite
DB <" + dbUrl + ">");
    }

    public static void initialize(String csvfile) {
        createTable();
        ArrayList<String[]> tpList = DbData.readFile(csvfile);
        for (int i = 0; i < tpList.size(); i++) {
            String[] tpItem = tpList.get(i);
            insert(tpItem);
        }
        JOptionPane.showMessageDialog(null, "SQLite DB <" + dbUrl + ">
Initialized");
    }

    public static String[] getList() {
        ArrayList<String> list = new ArrayList<String>();
        Connection conn = connect();
        if (conn == null) return null;
        String sql = "SELECT Tpid, LastName, FirstName FROM TennisPlayers";
        try {
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(sql);
            while (rs.next()) {
                String str = rs.getString("Tpid");
                str += ": " + rs.getString("LastName");
                str += ", " + rs.getString("FirstName");
                list.add(str);
            }
            String[] slist = new String[list.size()];
            int n = 0;
            for (String s : list) {
                slist[n] = s;
                n += 1;
            }
            st.close();
            rs.close();
            conn.close();
            JOptionPane.showMessageDialog(null, "SQLite DB <" + dbUrl + ">
Data List Obtained");
            return slist;
        }
        catch (Exception e) {
            System.out.println("<<< " + e.getMessage());
            JOptionPane.showMessageDialog(null, "<<< SELECT Operation to
SQLite DB <" + dbUrl + "> Failed");
            return null;
        }
    }

    public static void insert(String[] flist) {
        Connection conn = connect();
        if (conn == null) return;
        try {
            String sql = "INSERT INTO TennisPlayers(Tpid, LastName,
FirstName, Age, Country, Organization, Ranking, Points) VALUES(?, ?, ?, ?,
?, ?, ?, ?)";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setString(1, flist[TPF_TPID]);
            ps.setString(2, flist[TPF_LASTNAME]);
            ps.setString(3, flist[TPF_FIRSTNAME]);
            ps.setString(4, flist[TPF_AGE]);
            ps.setString(5, flist[TPF_COUNTRY]);
            ps.setString(6, flist[TPF_ORGANIZATION]);
            ps.setString(7, flist[TPF_RANKING]);
            ps.setString(8, flist[TPF_POINTS]);
            ps.executeUpdate();
            ps.close();
            conn.close();
        }
        catch (Exception e) {
            System.out.println("<<< " + e.getMessage());
            JOptionPane.showMessageDialog(null, "<<< Inserting Data in
SQLite DB <" + dbUrl + "> Failed");
            return;
        }
    }

}
```

26. The `sqliteDb` class defines the following methods:
    - setDb() - creates a DB name
    - connect() - connects the database
    - createTable() - create a data table in the database
    - initialize() - insert the initial data into the database
    - getList() - gets the list of entries in the database
    - insert() - inserts a data into the database

---

27. Go to DbApp1.java.

28. Add the following import statement after the last import statement.

```
import java.sql.*;
```

29. Add the following declaration at the beginning of the `DbApp1` class.

```
private JComboBox<String> comboBoxList;
```

30. Change the JCombobox declaration at the end of `initialize()`.

```
comboBoxList = new JComboBox<String>();
```

31. Add the following method before `initialize()`.

```
public void updateComboBoxList() {
    comboBoxList.removeAllItems();
    String[] list = SqliteDb.getList();
    for (String s : list) comboBoxList.addItem(s);
}
```

32. Go to Desinger.

33. Double-click on btnInit.

34. Add the following statements in `actionPerformed(ActionEvent e)` for `btnInit`.

```
SqliteDb.setDb(txtDbName.getText());
SqliteDb.initialize(txtCsvFile.getText());
updateComboBoxList();
```

35. Go to Designer.

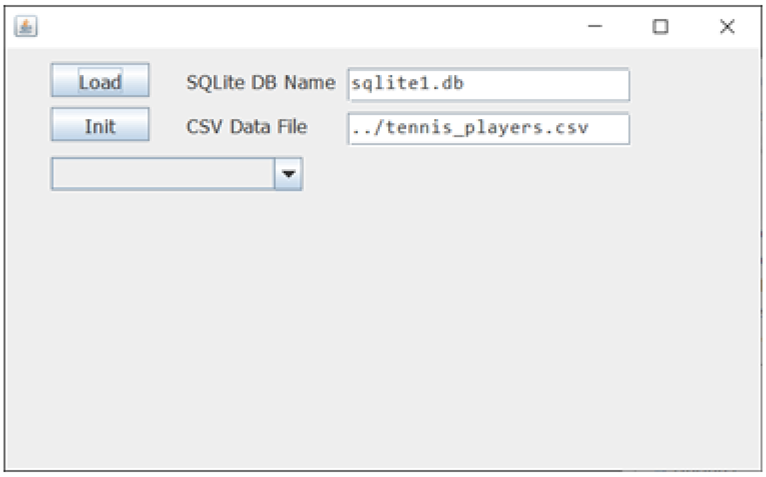
36. Double-click on btnLoad.

37. Add the following statements in `actionPerformed(ActionEvent e)` for `btnLoad`.

```
SqliteDb.setDb(txtDbName.getText());
updateComboBoxList();
```

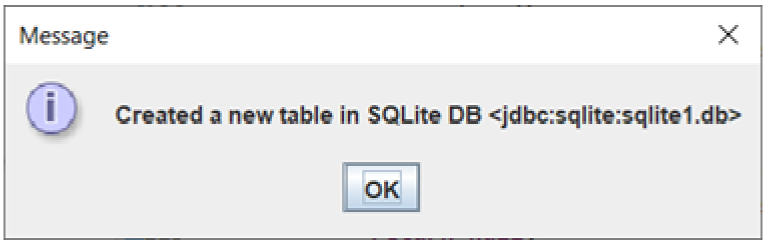
38. Run the program.

39. The following window should appear.

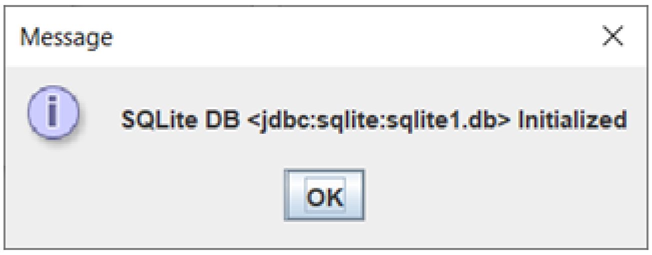


40. Click on Init.

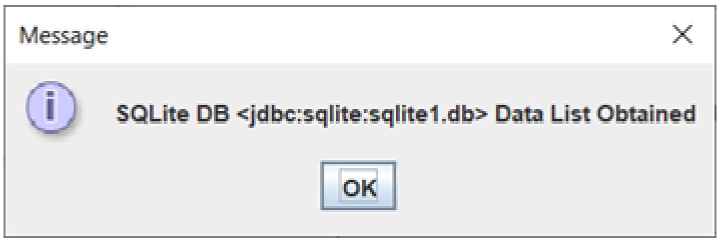
41. The following messages appear.



Created a new table in SQLite DB <jdbc:sqlite:sqlite1.db>

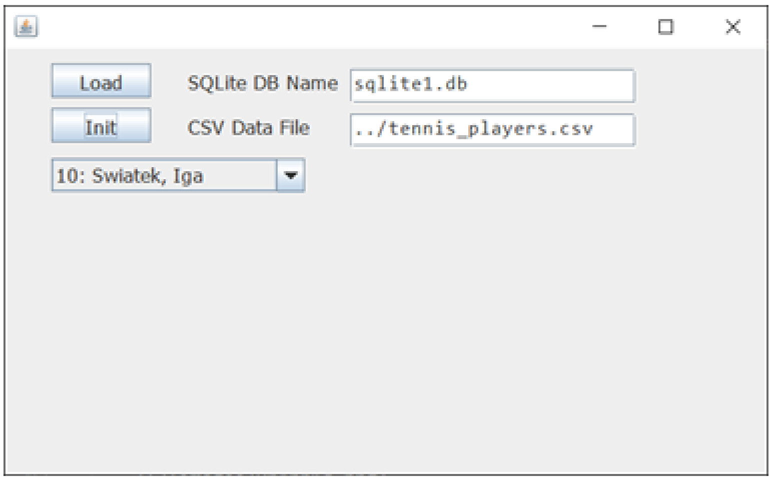


SQLite DB <jdbc:sqlite:sqlite1.db> Initialized



SQLite DB <jdbc:sqlite:sqlite1.db> Data List Obtained

42. The final windonw looks like:



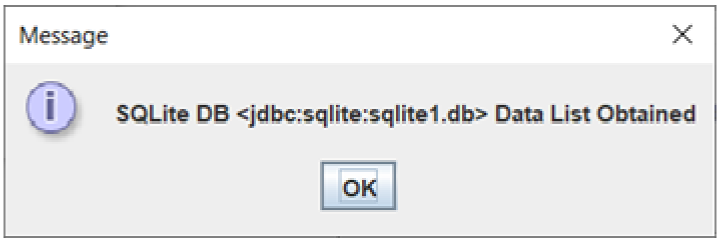
SQLite DB Name: sqlite1.db
CSV Data File: ../tennis_players.csv
10: Swiatek, Iga

43. Terminate the program.

44. Run the program.

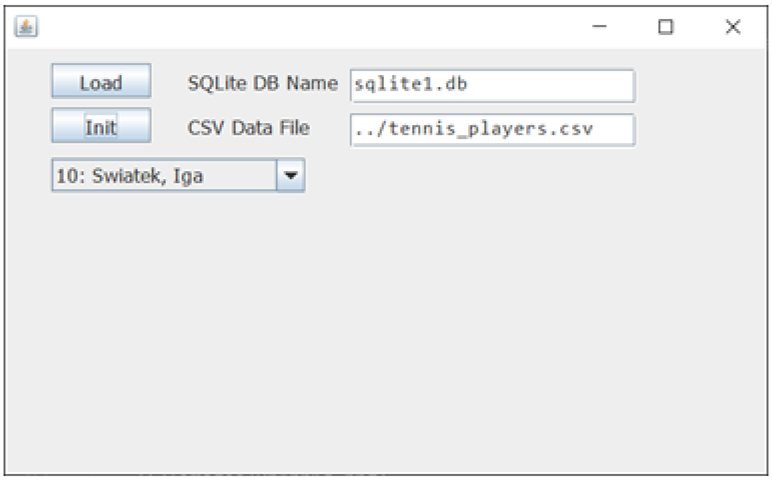
45. Click on Load.

46. The following message should appear.



SQLite DB <jdbc:sqlite:sqlite1.db> Data List Obtained

47. The final window looks like:



Note 1: The first click on Init creates a new database with the data read from "../tennis_players.csv".

Note 2: The second click on Load obtains the data list from the creaed database.

Note 3: The "splite1.db" file is created in the workspace.

Note 4: The click on the JComboBox shows the list of data stored in the database: Tpid: LastName, FirstName.

Note 5: `sqlite-jdbc-3.41.2.1.jar` is stored in Lab1_Files.

Note 6: `tennis_players.csv` is stored in Lab1_Files.

## Database Application Step 2

1. Create a new Java project "DbApp2".
2. Copy DbApp1/src/DbApp1.java to DbApp2/src.
3. Copy DbApp1/src/DbData.java to DbApp2/src.
4. Copy DbApp1/src/SqliteDb.java to DbApp2/src.
5. Rename DbApp2/src/DbApp1 to DbApp2.java by Refactor -> Rename.

---

6. Right-click on DbApp2 -> JRE System Library.
7. Build Path -> Configure Build Path...
8. Scoll up and click on Modulepath.
9. Click on Add External JARs.
10. Choose the downloaded sqlite-jdbc jar file.
11. Click on Apply and Close.

---

12. Add the following label-textbox pairs:

    - lblTpid (Text: Tpid) + txtTpid (Text: blank)
    - lblLastName (Text: LastName) + txtLastName
    - lblFirstName (Text: FirstName) + txtxFirstName
    - lblAge (Text: Age) + txtAge
    - lblCountry (Text: Country) + txtCountry
    - lblOrganization (Text: Organization) + txtOrganization
    - lblRanking (Text: Ranking) + txtRanking

- lblPoints (Text: Points) + txtPoints

13. They should look like:

14. Add the following two methods before `updateComboBoxList()` in the `DbApp2` class.

```
public String[] getTextFieldValues() {
    String[] str = new String[SqliteDb.TPF_COUNT];
    str[SqliteDb.TPF_TPID] = txtTpid.getText();
    str[SqliteDb.TPF_LASTNAME] = txtLastName.getText();
    str[SqliteDb.TPF_FIRSTNAME] = txtFirstName.getText();
    str[SqliteDb.TPF_AGE] = txtAge.getText();
    str[SqliteDb.TPF_COUNTRY] = txtCountry.getText();
    str[SqliteDb.TPF_ORGANIZATION] = txtOrganization.getText();
    str[SqliteDb.TPF_RANKING] = txtRanking.getText();
    str[SqliteDb.TPF_POINTS] = txtPoints.getText();
    return str;
}

public void setTextFieldValues(String[] flist) {
    txtTpid.setText(flist[SqliteDb.TPF_TPID]);
    txtLastName.setText(flist[SqliteDb.TPF_LASTNAME]);
    txtFirstName.setText(flist[SqliteDb.TPF_FIRSTNAME]);
    txtAge.setText(flist[SqliteDb.TPF_AGE]);
    txtCountry.setText(flist[SqliteDb.TPF_COUNTRY]);
    txtOrganization.setText(flist[SqliteDb.TPF_ORGANIZATION]);
    txtRanking.setText(flist[SqliteDb.TPF_RANKING]);
    txtPoints.setText(flist[SqliteDb.TPF_POINTS]);
}
```

15. Go to Designer.

16. Righ-click on `ComboBoxList` -> Add event handler -> action.

17. Go back to Souce.

18. Add the following code in `actionPerformed(ActionEvent e)` for `comboBoxList`.

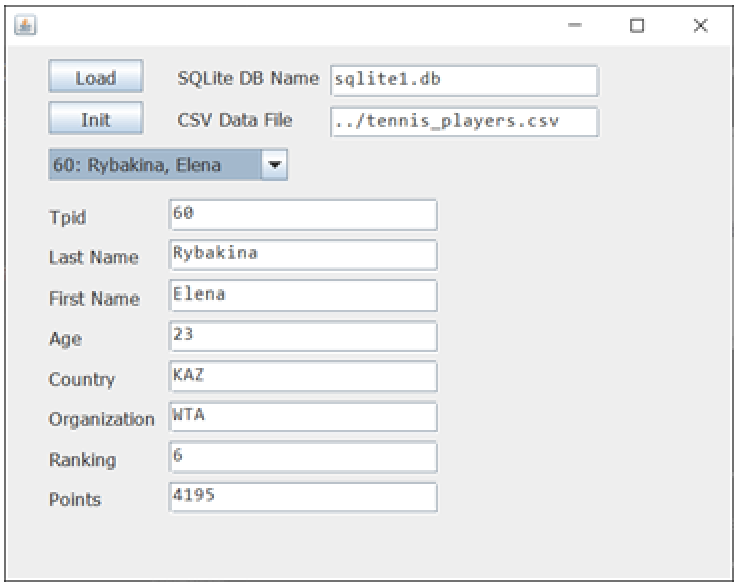
```
String selected = (String) comboBoxList.getSelectedItem();
if (selected == null) return;
String[] item = selected.split("[:]", 0);
ArrayList<String[]> vlist = SqliteDb.find(SqliteDb.TPF_TPID, item[0]);
setTextFieldValues(vlist.get(0));
```

19. Add the following method to the `SqliteDb` class.

```
public static ArrayList<String[]> find(int tpfx, String fval) {
    Connection conn = connect();
    if (conn == null) return null;
    try {
        String sql = "SELECT * FROM TennisPlayers WHERE " + TPF_NAME[tpfx] +
" = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, fval);
        ResultSet rs = ps.executeQuery();
        ArrayList<String[]> flist = new ArrayList<String[]>();
        while (rs.next()) {
            String[] str = new String[TPF_COUNT];
            for (int i = 0; i < TPF_COUNT; i++) str[i] =
rs.getString(TPF_NAME[i]);
            flist.add(str);
        }
        rs.close();
        ps.close();
        conn.close();
        return flist;
    }
    catch (Exception e) {
        System.out.println("<<< " + e.getMessage());
        JOptionPane.showMessageDialog(null, "<<< SELECT Operation to SQLite
DB <" + dbUrl + "> Failed");
        return null;
    }
}
```

20. Run the program.

21. Select one of entry in `ComboBoxList` and the data of the selected item are shown in the text fields.



| | |
|---|---|
| SQLite DB Name | sqlite1.db |
| CSV Data File | ../tennis_players.csv |
| Tpid | 60 |
| Last Name | Rybakina |
| First Name | Elena |
| Age | 23 |
| Country | KAZ |
| Organization | WTA |
| Ranking | 6 |
| Points | 4195 |

## Database Application Step 3

1. Create a new Java project "DbApp3".
2. Copy DbApp2/src/DbApp2.java to DbApp3/src.
3. Copy DbApp2/src/SqliteDb.java to DbApp3/src.
4. Copy DbApp2/src/DbData.java to DbAPp3/src.
5. Rename DbApp3/src/DbApp2.java to DbApp3.java by Refactor -> Rename.

---

6. Right-click on DbApp3 -> JRE System Library.
7. Build Path -> Configure Build Path...
8. Scoll up and click on Modulepath.
9. Click on Add External JARs.
10. Choose the downloaded sqlite-jdbc jar file.
11. Click on Apply and Close.

---

12. Add the `Clear` button (btnClear) in the `DbApp3` class.

13. Add the following code in its `actionPerformed()`.

```
btnClear.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String[] flist = {"", "", "", "", "", "", "", ""};
        setTextFieldValues(flist);
    }
});
```

14. The click on `Clear` should clear all the text fields.

---

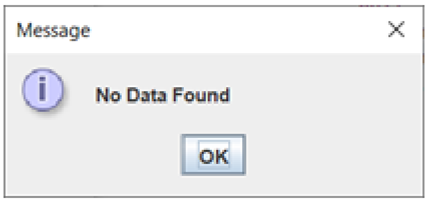
15. Add the `Find` button (btnFind) in the `DbApp3` class.

16. Add the following code in its `actionPerformed()`.

```
btnFind.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String[] flist = getTextFieldValues();
        ArrayList<String[]> vlist = null;
        for (int i = 0; i < SqliteDb.TPF_COUNT; i++) {
            if (flist[i].length() == 0) continue;
            vlist = SqliteDb.find(i,  flist[i]);
            break;
        }
        if (vlist.size() == 0) {
            JOptionPane.showMessageDialog(null, "No Data Found");
            return;
        }
        Object[] options = {"Yes", "No"};
        for (int i = 0; i < vlist.size(); i++) {
            setTextFieldValues(vlist.get(i));
            if (vlist.size() > 1) {
                int x = JOptionPane.showOptionDialog(null,
                            "Next Data?",
                            "Choose Yes/No",
                            JOptionPane.YES_NO_OPTION,
                            JOptionPane.QUESTION_MESSAGE,
                            null,
                            options,
                            options[0]);
                if (x == 1) break;
            }
        }
    }
});
```

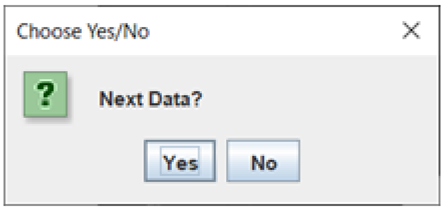
17. The click on `Find` takes the non-blank field item and finds the data with the field item.

18. If it finds one, it shows the data in the text fields. If it does not find the data, it shows the message.



No Data Found

19. If it finds multiple data, then it shows one at a time with the following message.



Next Data?

---

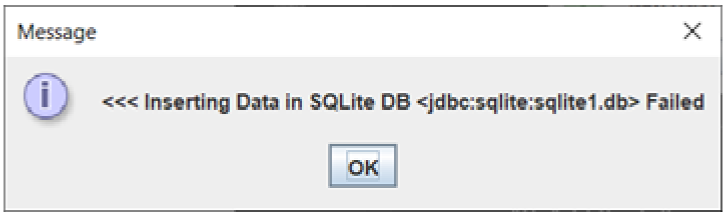
20. Add the `Insert` button (btnInsert) in the `DbApp3` class.

21. Add the following code in its `actionPerformed()`.

```
btnInsert.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String[] flist = getTextFieldValues();
        SqliteDb.insert(flist);
        updateComboBoxList();
    }
});
```

22. The click on `Insert` inserts the new data with the items in the text fields into the database and updates the ComboBoxList.

23. If the data with the Tpid item already exists in the database, the insert operation fails.



<<< Inserting Data in SQLite DB <jdbc:sqlite:sqlite1.db> Failed

---

24. Add the `Update` button (btnUpdate) in the `DbApp3` class.

25. Add the following code in its `actionPerformed()`.

```
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String[] flist = getTextFieldValues();
        SqliteDb.update(flist);
        updateComboBoxList();
    }
});
```

26. Add the following method in the `SqliteDb` class.

```
public static void update(String[] flist) {
    Connection conn = connect();
    if (conn == null) return;
    try {
        String sql = "UPDATE TennisPlayers SET LastName = ?, FirstName = ?,
Age = ?, Country = ?, Organization = ?, Ranking = ?, Points = ? WHERE Tpid =
?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, flist[TPF_LASTNAME]);
        ps.setString(2, flist[TPF_FIRSTNAME]);
        ps.setString(3, flist[TPF_AGE]);
        ps.setString(4, flist[TPF_COUNTRY]);
        ps.setString(5, flist[TPF_ORGANIZATION]);
        ps.setString(6, flist[TPF_RANKING]);
        ps.setString(7, flist[TPF_POINTS]);
        ps.setString(8, flist[TPF_TPID]);
        ps.executeUpdate();
        ps.close();
        conn.close();
    }
    catch (Exception e) {
        System.out.println("<<< " + e.getMessage());
        JOptionPane.showMessageDialog(null, "<<< Updating Data in SQLite DB
<" + dbUrl + "> Failed");
        return;
    }
}
```

27. The click on `Update` updates the data with the new items in one or more text fields and also updates the ComboBoxList.

---

28. Add the `Delete` button (btnDelete) in the `DbApp3` class.

29. Add the following code in its `actionPerformed()`.

```
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        SqliteDb.delete(txtTpid.getText());
        updateComboBoxList();
    }
});
```

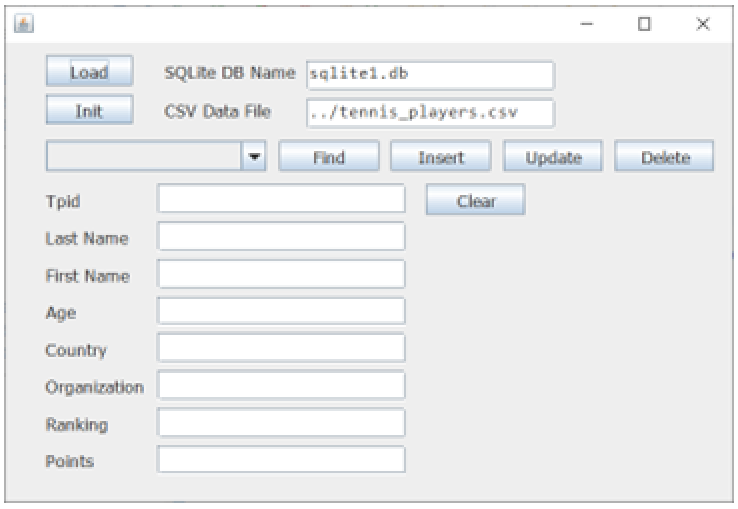
30. Add the following method in the `SqliteDb` class.

```
public static void delete(String tpid) {
    Connection conn = connect();
    if (conn == null) return;
    try {
        String sql = "DELETE FROM TennisPlayers WHERE Tpid = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, tpid);
        ps.executeUpdate();
        ps.close();
        conn.close();
    }
    catch (Exception e) {
        System.out.println("<<< " + e.getMessage());
        JOptionPane.showMessageDialog(null, "<<< Deleting Data in SQLite DB
<" + dbUrl + "> Failed");
        return;
    }
}
```

31. The click on `Delete` deletes the data with the specified Tpid.

---

32. The window with the `Find`, `Insert`, `Update`, `Delete`, and `Clear` buttons looks like:



---

33. Add the new label `lblImage` in the `DbApp3` class.

34. Add the following import statement.

```
import javax.swing.ImageIcon;
```

35. Add the following code at the beginning of the `DbApp3` class.

```
private JLabel lblImage;
private int imgNum = 0;
```

36. Change the `lblImage` statement if necessary.

```
lblImage = new JLabel("Image");
```

37. Change the instantiation of JLabel for image at the end of `initialize()`.

```
lblImage = new JLabel("Image");
lblImage.setBounds(419, 205, 100, 140);
ImageIcon img = new ImageIcon("../tennis_player_3_100x137.jpg");
lblImage.setIcon(img);
frame.getContentPane().add(lblImage);
imgNum = 0;
```

38. Add the following method in the `DbApp3` class.

```
public class ImageThread extends Thread {
    public void run() {
        while (true) {
            try {
                ImageIcon img = null;
                sleep(10000); // 10,000 ms
                switch (imgNum) {
                case 0: img = new
ImageIcon("../tennis_player_2_100x139.jpg"); break;
                case 1: img = new
ImageIcon("../tennis_player_3_100x137.jpg"); break;
                case 2: img = new
ImageIcon("../tennis_player_1_100x138.jpg"); break;
                }
                lblImage.setIcon(img);
                imgNum += 1;
                if (imgNum > 2) imgNum = 0;
            }
            catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

39. Add the following code at the end of `initialize()`.

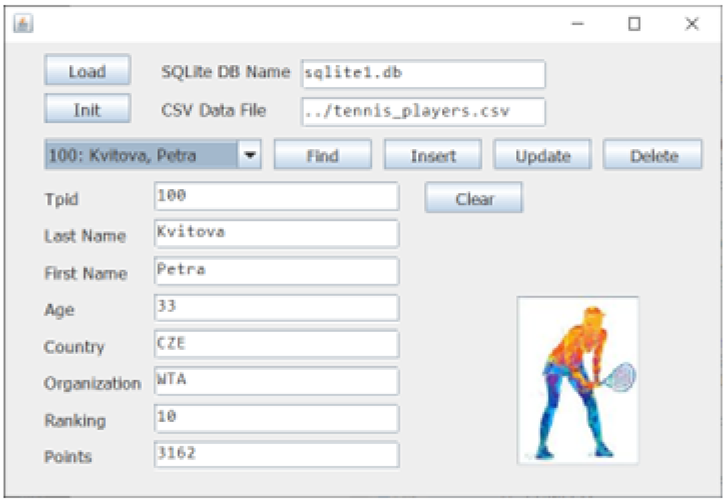
```
ImageThread it = new ImageThread();
it.start();
```

40. Place the three tennis player images in the Java workplace.

    - tennis_player_1_100x138.jpg
    - tennis_player_2_100x139.jpg
    - tennis_player_3_100x137.jpg

41. Run the program.

42. Check that three tennis player images change every 10 sec.



SQLite DB Name: sqlite1.db
CSV Data File: ../tennis_players.csv
100: Kvitova, Petra

| | |
|---|---|
| Tpid | 100 |
| Last Name | Kvitova |
| First Name | Petra |
| Age | 33 |
| Country | CZE |
| Organization | WTA |
| Ranking | 10 |
| Points | 3162 |

---

END OF Lab 1