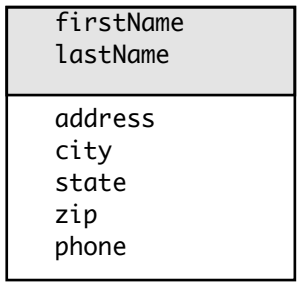
CPS221 - SOFTWARE SYSTEMS

An Example of a JDBC Application: Address Book Demonstration Revised to use a Database

(revised 11/3/11)

To revise the address book we used for demonstrations in CPS122 to store the address book in a database, rather than in an ordinary disk file, we could make the following changes:

- Create a one-table database with the following schema:



- Eliminate the `AddressBook, FileSystem` and `Person` classes (no longer needed, since the address book is being kept in a database).

- Add a new class for representing the full name of a person. In the database version, the full name serves as the key for persons, replacing the notion of object identity in the OO version.

- Eliminate the `New, Open, Save,` and `Save As...` options from the `File` menu, along with related functionality (e.g. keeping track of unsaved changes, offering to save unsaved changes, graying out the `Save` option). (These are no longer needed).

- Add an instance variable to keep track of whether the `ORDER BY` clause in `SELECT` statements should specify Name order or ZIP order. (Note a database table, as a set, has no inherent order - we determine the order we want when we access it rather than by sorting it - thus the "sort" buttons simply change the value of this variable.)

We also need to change the way the user interface is structured, but this does not concern us here.

Code for the revised `AddressBookController` is given below. Other needed changes (e.g deleting unneeded menu options and related functionality and modifying the way the list of names is displayed in the GUI; the new FullName class, changes to the top-level application and applet) are omitted.

This handout is based on an earlier version of the address book example, so the code does not exactly correspond to the current version - but the basic idea is the same.

Note: the code below optionally prints out each SQL statement and its results to `System.out`*, for debugging and demonstration purposes. This can be turned on or off by changing the value of the constant LOG_SQL at the end of the* `AddressBookController` *class. Obviously, in real production code, this logging would not be done!* ***Students using this code as an example for work should not include any of this code!***

```
                                    // IMPORT FOR java.io.File NO LONGER NEEDED
import java.io.IOException;
import javax.swing.JOptionPane;
                          // IMPORT FOR javax.swing.JFileChooser NO LONGER NEEDED
import javax.swing.DefaultListModel;                                    // ADDED
import java.sql.*;                                                      // ADDED

/** An object of this class performs operations on the address book in
 *  response to user gestures on the GUI
 *
 */
public class AddressBookController
{
    /** Constructor
     *                                  // fileSystem PARAMETER NO LONGER NEEDED
     */
    public AddressBookController() {
        // Ensure that the needed database driver is loaded             // ADDED
                                                                        //   |
        try                                                             //   |
        {                                                               //   |
            Class.forName(DRIVER_NAME);                                 //   |
        }                                                               //   |
        catch(ClassNotFoundException e)                                 //   |
        {                                                               //   |
            System.out.println("DBMS driver not found");                 //   |
            System.exit(1);                                             //   |
        }                                                               //   |
                                                                        //   |
        // Connect to the database, and create a statement object that  //   |
        // all methods will share                                       //   |
                                                                        //   |
        try                                                             //   |
        {                                                               //   |
            connection = DriverManager.getConnection(                   //   |
                DATABASE_URL, DATABASE_USER, DATABASE_PASSWORD);        //   |
            statement = connection.createStatement();                   //   |
        }                                                               //   |
        catch (Exception e)                                             //   |
        {                                                               //   |
            System.out.println(                                         //   |
                "Connection failed : " + e.getMessage());               //   |
            e.printStackTrace();                                        //   |
            System.exit(1);                                             //   |
        }                                                               //   |

                    // AddressBook PARAMETER TO GUI CONSTRUCTOR NO LONGER NEEDED
        AddressBookGUI gui = new AddressBookGUI(this);

        gui.setVisible(true);
    }
```

```
/** Do the Add a Person Use Case.
 *
 *  @param gui the gui that requested this operation
 *  @return true if an add was done; false if it was cancelled      // ADDED
 */
public boolean doAdd(AddressBookGUI gui) {
    // Set up for the the dialog box

    String [] fieldNames = { "First Name", "Last Name", "Address",
                             "City", "State", "ZIP", "Phone" };

    // Get new person from the user

    String [] info = MultiInputPane.showMultiInputDialog(
        gui, fieldNames, "Person to add");

    // Add new person to book and update GUI, unless canceled

    if (info != null)
    {
        // Add the new person to the database               // THIS CODE
                                                            // REPLACES THE
        int rowCount = 0;                                   // CODE THAT
        try                                                 // CREATES A NEW
        {                                                   // Person OBJECT
            String sqlCommand =                             // AND ADDS IT
                "INSERT INTO ABTABLE VALUES('"+             // TO THE CURRENT
                    info[0] + "', '" +                      // ADDRESS BOOK
                    info[1] + "', '" +                      //       |
                    info[2] + "', '" +                      //       |
                    info[3] + "', '" +                      //       |
                    info[4] + "', '" +                      //       |
                    info[5] + "', '" +                      //       |
                    info[6] + "')";                         //       |
            if (LOG_SQL) System.out.println(sqlCommand);    //       |
            rowCount = statement.executeUpdate(sqlCommand); //       |
            if (LOG_SQL) System.out.println(                //       |
                "Rows affected " + rowCount);               //       |
        }                                                   //       |
        catch(SQLException e)                               //  EXCEPTION
        {                                                   //  HANDLING
            gui.reportError("Error: " + e);                 //  CODE FOR SQL
        }                                                   //       |

        // CODE TO MARK THE ADDRESS BOOK AS CHANGED AND ENABLE THE
        // SAVE MENU OPTION IS NO LONGER NEEDED

        return true;
    }
    else
        return false;
}
```

```
/** Do the Edit a Person use case
 *  @param gui the gui that requested this operation
 *  @param selected name the full name of the desired person // CHANGED TYPE
 *  @return true if an edit was done; false if it was cancelled     // ADDED
 */
public boolean doEdit(AddressBookGUI gui, FullName selectedName) {
    String [] initialValues = new String [5];
    ResultSet results = null;                                  // THIS CODE
    int rowCount = 0;                                          // REPLACES THE
    try                                                        // CODE THAT
    {                                                          // LOOKS UP THE
        String sqlQuery =                                      // PERSON'S INFO
            "SELECT ADDRESS, CITY, STATE, ZIP, PHONE " +       // IN THE CURRENT
                "FROM ABTABLE " +                              // ADDRESS BOOK
                "WHERE FIRSTNAME = '" +                        //       |
                    selectedName.getFirstName() + "'" +        //       |
                "AND LASTNAME = '" +                           //       |
                    selectedName.getLastName() + "'";          //       |
        if (LOG_SQL) System.out.println(sqlQuery);             //       |
        results = statement.executeQuery(sqlQuery);            //       |
        if (! results.next()) {  // This should not happen     //       |
            gui.reportError("Name not found");                 //       |
            return false;                                      //       |
        }                                                      //       |
        for (int i = 1; i <= 5; i ++)                          //       |
            initialValues[i-1] = results.getString(i);         //       |

        // Set up for the the dialog box
        String [] fieldNames = { "Address", "City", "State", "ZIP", "Phone" };

        // Get updated information from the user
        String [] info = MultiInputPane.showMultiInputDialog(
            gui,
            fieldNames,
            initialValues,
            "Edit " + selectedName);

        // Modify person in book, unless canceled

        if (info != null) {
            // Update the entry in the database                // THIS CODE
                                                               // REPLACES THE
            String sqlCommand =                                // CODE TO
                "UPDATE ABTABLE " +                            // MODIFY THE
                    "SET " +                                   // PERSON OBJECT
                        "ADDRESS = '" + info[0] + "', " +      // USING
                        "CITY = '" + info[1] + "', " +         // updatePerson()
                        "STATE = '" + info[2] + "', " +        //       |
                        "ZIP = '" + info[3] + "', " +          //       |
                        "PHONE = '" + info[4] + "' " +         //       |
                "WHERE FIRSTNAME = '" +                        //       |
                    selectedName.getFirstName() + "'" +        //       |
                "AND LASTNAME = '" +                           //       |
                    selectedName.getLastName() + "'";          //       |
            if (LOG_SQL) System.out.println(sqlCommand);       //       |
            rowCount = statement.executeUpdate(sqlCommand);    //       |
            if (LOG_SQL) System.out.println(                   //       |
                "Rows affected " + rowCount);                  //       |
            return true;                                       //       |
        }                                                      //       |
```

```
        else                                                    //      |
            return false;                                       //      |
    }                                                           //      |
    catch(SQLException e)                                       // EXCEPTION
    {                                                           // HANDLING
        gui.reportError("Error: " + e);                         // CODE FOR SQL
        return false;                                           //      |
    }                                                           //      |
    // CODE TO MARK THE ADDRESS BOOK AS CHANGED AND ENABLE THE
    // SAVE MENU OPTION IS NO LONGER NEEDED
}

/** Do the Delete a Person use case
 *  @param gui the gui that requested this operation
 *  @param selected name the full name of the desired person // CHANGED TYPE
 *  @return true if an edit was done; false if it was cancelled     // ADDED
 */
public boolean doDelete(AddressBookGUI gui, FullName selectedName)
{
    // Ask user to confirm
    if (JOptionPane.showConfirmDialog(
                gui,
                "Are you sure you want to delete " + selectedName + "?",
                "Confirm delete",
                JOptionPane.YES_NO_OPTION)
            == JOptionPane.YES_OPTION)
    {
        // Delete the person - print message if this fails
        int rowCount = 0;                                       // THIS CODE
        try                                                     // REPLACES THE
        {                                                       // CODE THAT
            String sqlCommand =                                 // CALLS THE
                "DELETE FROM ABTABLE " +                        // delete()
                    "WHERE FIRSTNAME = '" +                     // METHOD OF THE
                        selectedName.getFirstName() + "'" +     // CURRENT
                    "AND LASTNAME = '" +                        // ADDRESS BOOK
                        selectedName.getLastName() + "'";       //      |
            if (LOG_SQL) System.out.println(sqlCommand);        //      |
            rowCount = statement.executeUpdate(sqlCommand);     //      |
            if (LOG_SQL) System.out.println(                    //      |
                    "Rows affected " + rowCount);               //      |
                                                                //      |
            return true;                                        //      |
        }                                                       //      |
        catch(SQLException e)                                   // EXCEPTION
        {                                                       // HANDLING
            gui.reportError("Error: " + e);                     // CODE FOR SQL
            return false;                                       //      |
        }                                                       //      |
    }                                                           //      |
    else                                                        //      |
        return false;                                           //      |

                // CODE TO MARK THE ADDRESS BOOK AS CHANGED AND ENABLE THE
                // SAVE MENU OPTION IS NO LONGER NEEDED
}
```

```
/** Do the Sort Entries by Name Use Case
  *
  *  @param gui the gui that requested this operation
  */
 public void doSortByName(AddressBookGUI gui)
 {
     // THE APPROACH TAKEN BY THIS METHOD IS QUITE DIFFERENT IN THE JDBC
     // VERSION.  A DATABASE TABLE IS A SET - HENCE HAS NO NOTION OF THE ORDER
     // OF ITS ELEMENTS.  INSTEAD, WE GET THE EFFECT OF SORTING WHEN WE
     // USE AN ORDER BY CLAUSE IN A SELECT STATEMENT.  THIS METHOD SIMPLY
     // ESTABLISHES THE CORRECT ORDER TO USE.  THE GUI THEN UPDATES THE
     // DISPLAYED LIST TO REFLECT THE RESULTS

     sortOrder = "ORDER BY LASTNAME, FIRSTNAME";
 }

 /** Do the Sort Entries by ZIP Use Case
  *
  *  @param gui the gui that requested this operation
  */
 public void doSortByZip(AddressBookGUI gui) {
     // SEE COMMENT ON doSortByName()

     sortOrder = "ORDER BY ZIP";
 }

 /** Do the Print Entries Use Case
  *
  *  @param gui the gui that requested this operation
  */
 public void doPrint(AddressBookGUI gui) {
     try                                                    // THIS CODE
     {                                                      // REPLACES THE
         String sqlQuery =                                  // CODE THAT WAS
             "SELECT FIRSTNAME, LASTNAME, " +               // PART OF THE
                 "ADDRESS, CITY, STATE, ZIP, " +            // AddressBook
                 "PHONE FROM ABTABLE " + sortOrder;         // CLASS THAT
         if (LOG_SQL) System.out.println(sqlQuery);         // NO LONGER
         ResultSet results =                                // EXISTS
             statement.executeQuery(sqlQuery);              //      |
         while (results.next())                             //      |
         {                                                  //      |
             System.out.println(results.getString(1) +      //      |
                 " " + results.getString(2));               //      |
             System.out.println(results.getString(3));      //      |
             System.out.println(results.getString(4) +      //      |
                 ", " + results.getString(5) +              //      |
                 "  " + results.getString(6));              //      |
             System.out.println(results.getString(7));      //      |
             System.out.println();                          //      |
         }                                                  //      |
     }                                                      //      |
     catch(SQLException e)                                  // EXCEPTION
     {                                                      // HANDLING
         gui.reportError("Error: " + e);                    // CODE FOR SQL
     }                                                      //      |
 }
```

```
    /** Do the Quit Program use case
     *  @param gui the gui that requested this operation
     */
    public void doQuit(AddressBookGUI gui) {
        try {                                                   // AT QUIT, THE CONNECTION
            connection.close();                                 // TO THE DATABASE MUST BE
        }                                                       // CLOSED
        catch(SQLException e) {                                 // EXCEPTION
            gui.reportError("Error: " + e);                     // HANDLING
        }                                                       // CODE FOR SQL
        System.exit(0);
    }

    /** Fill in a name list with the full names of persons currently    // NEW METHOD
     *  address book                                                    //   |
     *  @param nameList the name list to fill in                        //   |
     */                                                                 //   |
    public void fillIn(DefaultListModel nameList) {                     //   |
        ResultSet results;                                              //   |
        try {                                                           //   |
            String sqlQuery =                                           //   |
                "SELECT FIRSTNAME, LASTNAME FROM ABTABLE " + sortOrder; //   |
            if (LOG_SQL) System.out.println(sqlQuery);                  //   |
            results = statement.executeQuery(sqlQuery);                 //   |
            while(results.next())                                       //   |
                nameList.addElement(                                    //   |
                    new FullName(results.getString(1),                  //   |
                                 results.getString(2)));                //   |
        }                                                               //   |
        catch (SQLException e) {                                        //   |
            System.out.println(                                         //   |
                "Error looking up names" + e.getMessage());             //   |
            e.printStackTrace();                                        //   |
        }                                                               //   |
    }
                            // CODE FOR NEW, OPEN, SAVE, SAVE AS USE CASES PLUS
                            // OFFER TO SAVE CHANGES EXTENSION NO LONGER NEEDED

    private Connection connection;                                      // ADDED
    private Statement statement;                                        // ADDED
    private String sortOrder = "ORDER BY LASTNAME, FIRSTNAME";          // ADDED
                               // INSTANCE VARIABLE fileSystem NO LONGER NEEDED

    // The following was added for debugging and classroom demonstration
    // purposes.    When true, all SQL queries are logged to System.out.
    // When false, no logging occurs.  STUDENTS SHOULD NOT INCLUDE THIS CODE
    // (OR CODE THAT REFERENCES LOG_SQL) IN WORK THEY DO - IT IS FOR
    // DEMONSTRATION PURPOSES ONLY!
    private static final boolean LOG_SQL = false;                       // ADDED

    // These constants define the database we are using
    private static final String                                         // ADDED
        DRIVER_NAME = "org.gjt.mm.mysql.Driver",                        //   |
        DATABASE_URL =                                                  //   |
           "jdbc:mysql://dbms.cs.gordon.edu:3306/ADDRESS_BOOK",         //   |
        DATABASE_USER = "cps221",                                       //   |
        DATABASE_PASSWORD = "javarules";                                //   |
}
```