CST8288 OOP with Design Patterns

Data Access Object (DAO) Design Pattern

What is Data Access Object (DAO) Design Pattern?

- Common approach to architecting Data Connected Applications
- Separation of data persistence logic from rest of program
    - Allows for easier testing, maintenance, and updates
    - Also allows for easier reuse with other projects
    - Simple example presented in this course
      (can get much more complex, see recommended readings)

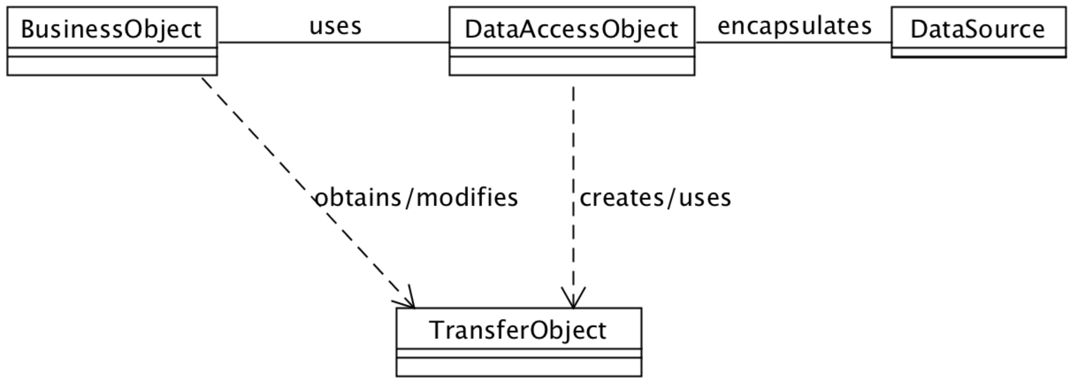


Figure 1.0 DAO design pattern. Note that the dashed line indicates a dependency. Diagram taken from Oracle (n.d.).

You might want to review Dependency, Association, Aggregation, Composition via this reading:
Niraj Bhatt. (July 15, 2011). Association vs. Dependency vs. Aggregation vs. Composition. Retrieved from https://nirajrules.wordpress.com/2011/07/15/association-vs-dependency-vs-aggregation-vs-composition/

DataSource

- Database, flat file, XML file etc. (Oracle n.d.)

DataAccessObject

- Focus of the pattern, is concerned with how the data gets stored, Oracle (n.d.)
- In our course we will place JDBC with SQL statements inside the DataAccessObject
- We will also make some public methods that allow the BusinessObject to get and send TransferObjects

BusinessObject

- Client code that needs to access and store data in the DataSource, Oracle (n.d.)

TransferObject

- Carries the data (Oracle n.d.).
- In our case this represents one row of data from a database table.
- Note: Older versions of Java's documentation confused Value Object Pattern with Transfer Object Pattern. The correct term for the pattern above is a Transfer Object. See recommended readings for more information on Value Object vs TransferObject (e.g. Primitive wrapper classes in Java are Value objects)

Sample Implementation of DAO based on works of Ram N. (2013 a, b, c, d)

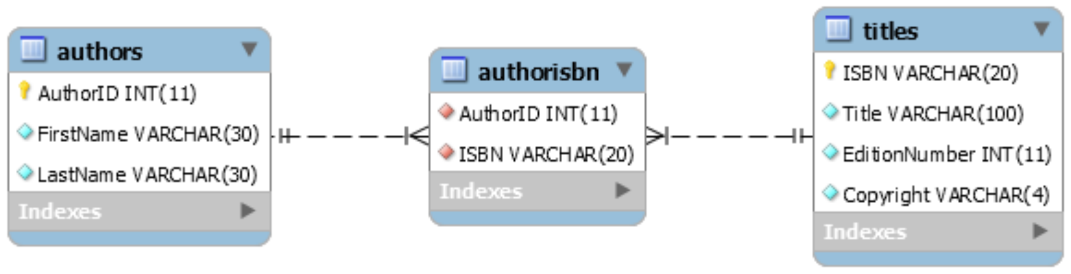


Figure 1.0 Database schema, books database, authors table, from textbook (Deitel and Deitel 10th edition)

DAO Part

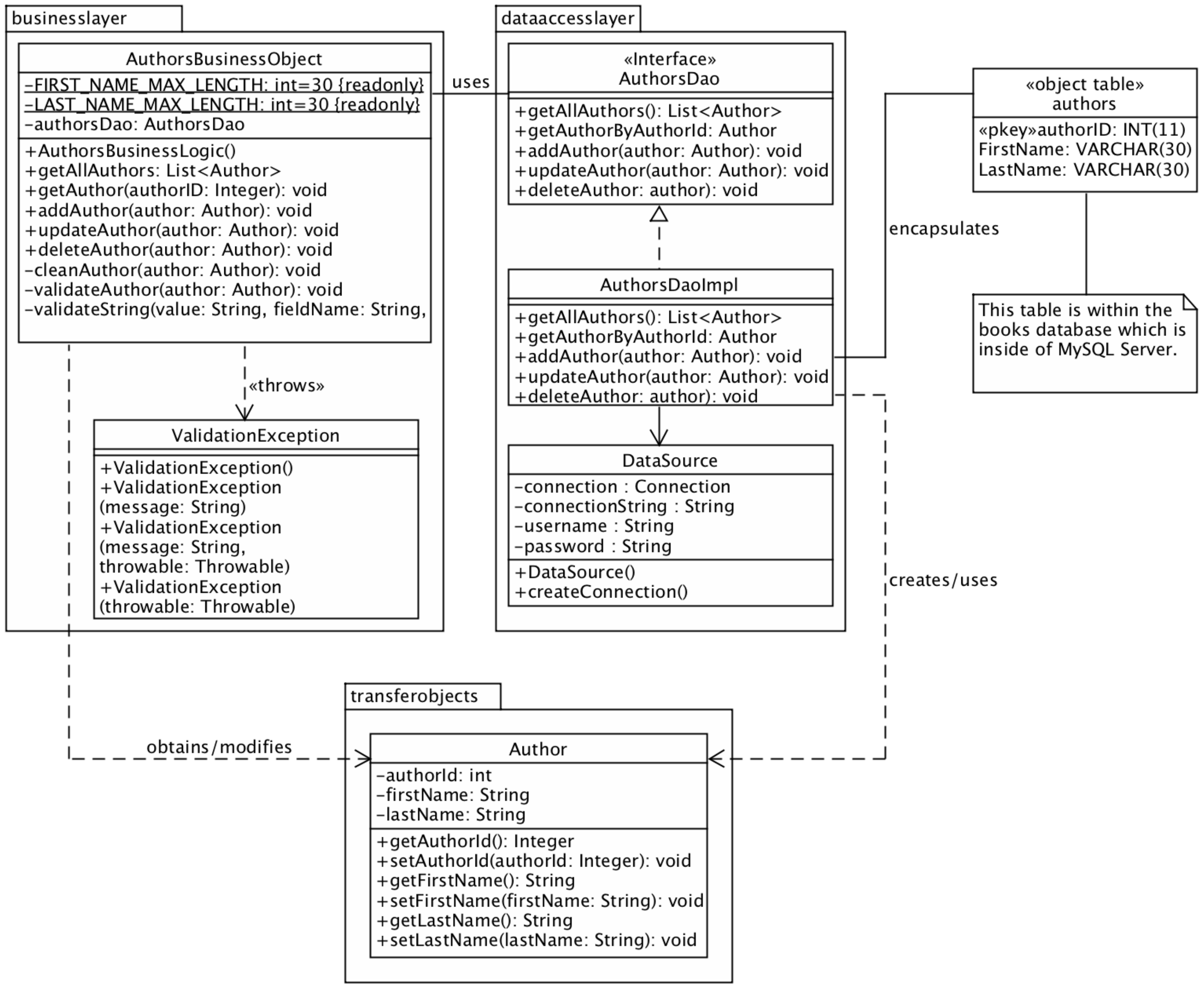

businesslayer
AuthorsBusinessObject
–FIRST_NAME_MAX_LENGTH: int=30 {readonly}
–LAST_NAME_MAX_LENGTH: int=30 {readonly}
–authorsDao: AuthorsDao
+AuthorsBusinessLogic()
+getAllAuthors: List<Author>
+getAuthor(authorID: Integer): void
+addAuthor(author: Author): void
+updateAuthor(author: Author): void
+deleteAuthor(author: Author): void
–cleanAuthor(author: Author): void
–validateAuthor(author: Author): void
–validateString(value: String, fieldName: String,
«throws»
ValidationException
+ValidationException()
+ValidationException
(message: String)
+ValidationException
(message: String,
throwable: Throwable)
+ValidationException
(throwable: Throwable)
uses
dataaccesslayer
«Interface»
AuthorsDao
+getAllAuthors(): List<Author>
+getAuthorByAuthorId: Author
+addAuthor(author: Author): void
+updateAuthor(author: Author): void
+deleteAuthor: author): void
AuthorsDaoImpl
+getAllAuthors(): List<Author>
+getAuthorByAuthorId: Author
+addAuthor(author: Author): void
+updateAuthor(author: Author): void
+deleteAuthor: author): void
DataSource
–connection : Connection
–connectionString : String
–username : String
–password : String
+DataSource()
+createConnection()
«object table»
authors
«pkey»authorID: INT(11)
FirstName: VARCHAR(30)
LastName: VARCHAR(30)
encapsulates
This table is within the books database which is inside of MySQL Server.
creates/uses
obtains/modifies
transferobjects
Author
–authorId: int
–firstName: String
–lastName: String
+getAuthorId(): Integer
+setAuthorId(authorId: Integer): void
+getFirstName(): String
+setFirstName(firstName: String): void
+getLastName(): String
+setLastName(lastName: String): void

Figure 2.0 UML for ADO design pattern, follows after work by Ram N. (2013 a, b, c, d)

Sample of a View with MVC

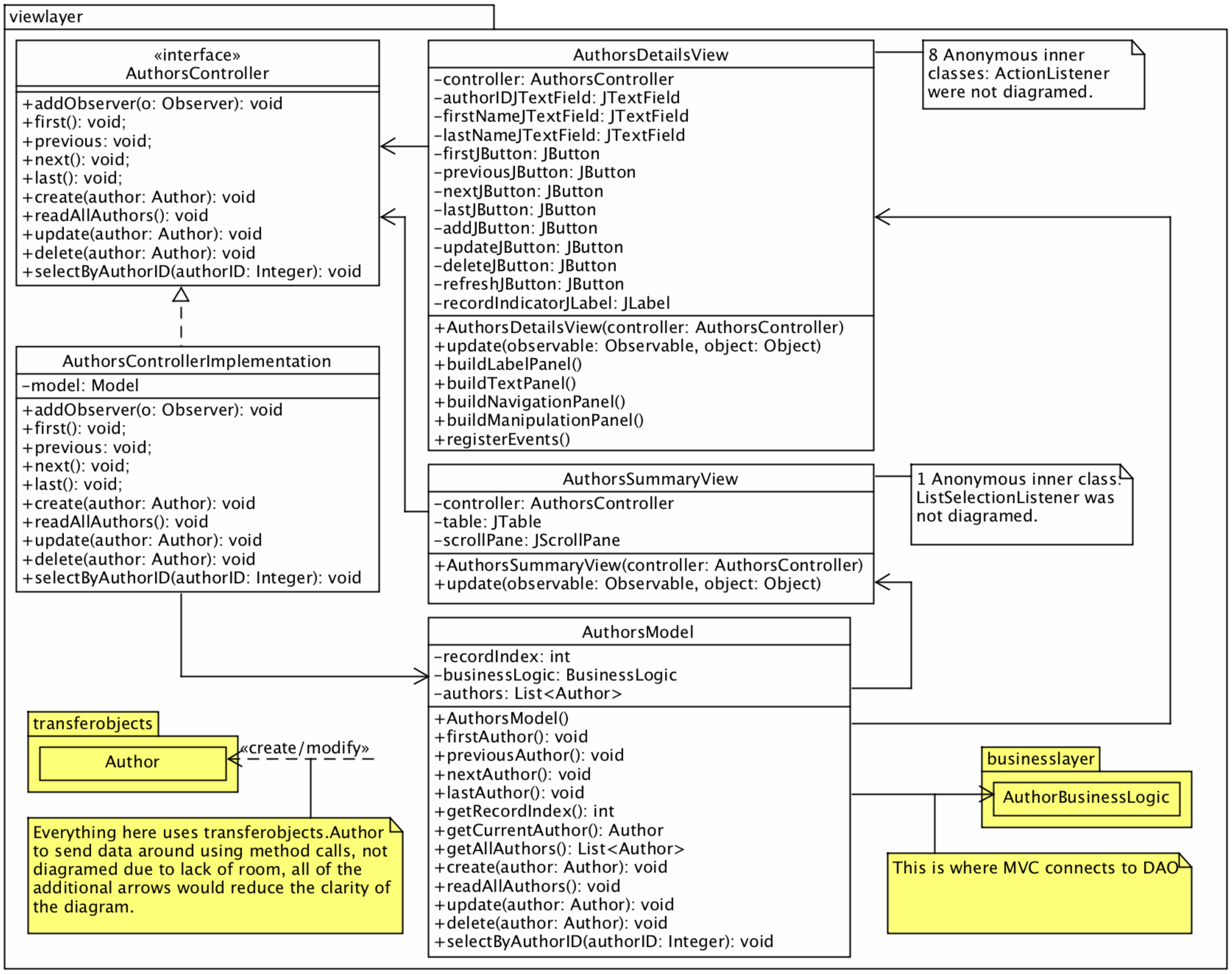
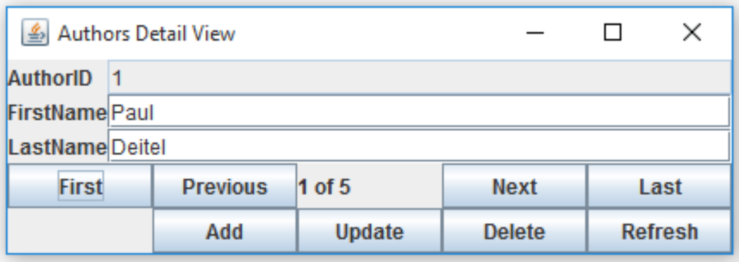

viewlayer
«interface»
AuthorsController
+addObserver(o: Observer): void
+first(): void;
+previous: void;
+next(): void;
+last(): void;
+create(author: Author): void
+readAllAuthors(): void
+update(author: Author): void
+delete(author: Author): void
+selectByAuthorID(authorID: Integer): void
AuthorsControllerImplementation
−model: Model
+addObserver(o: Observer): void
+first(): void;
+previous: void;
+next(): void;
+last(): void;
+create(author: Author): void
+readAllAuthors(): void
+update(author: Author): void
+delete(author: Author): void
+selectByAuthorID(authorID: Integer): void
AuthorsDetailsView
−controller: AuthorsController
−authorIDJTextField: JTextField
−firstNameJTextField: JTextField
−lastNameJTextField: JTextField
−firstJButton: JButton
−previousJButton: JButton
−nextJButton: JButton
−lastJButton: JButton
−addJButton: JButton
−updateJButton: JButton
−deleteJButton: JButton
−refreshJButton: JButton
−recordIndicatorJLabel: JLabel
+AuthorsDetailsView(controller: AuthorsController)
+update(observable: Observable, object: Object)
+buildLabelPanel()
+buildTextPanel()
+buildNavigationPanel()
+buildManipulationPanel()
+registerEvents()
8 Anonymous inner classes: ActionListener were not diagramed.
AuthorsSummaryView
−controller: AuthorsController
−table: JTable
−scrollPane: JScrollPane
+AuthorsSummaryView(controller: AuthorsController)
+update(observable: Observable, object: Object)
1 Anonymous inner class: ListSelectionListener was not diagramed.
AuthorsModel
−recordIndex: int
−businessLogic: BusinessLogic
−authors: List<Author>
+AuthorsModel()
+firstAuthor(): void
+previousAuthor(): void
+nextAuthor(): void
+lastAuthor(): void
+getRecordIndex(): int
+getCurrentAuthor(): Author
+getAllAuthors(): List<Author>
+create(author: Author): void
+readAllAuthors(): void
+update(author: Author): void
+delete(author: Author): void
+selectByAuthorID(authorID: Integer): void
transferobjects
Author
«create/modify»
Everything here uses transferobjects.Author to send data around using method calls, not diagramed due to lack of room, all of the additional arrows would reduce the clarity of the diagram.
businesslayer
AuthorBusinessLogic
This is where MVC connects to DAO

Figure 3.0 UML for MVC of view layer

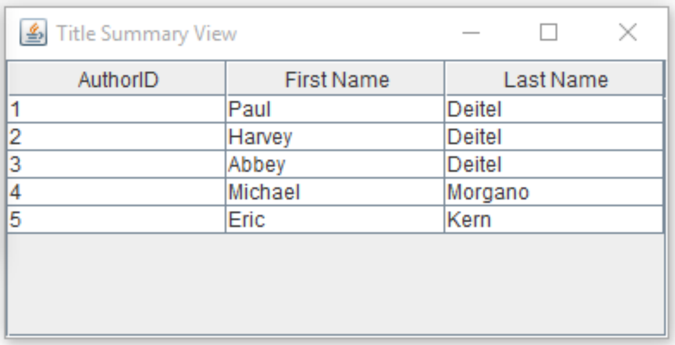
Demo with code walk-through


Authors Detail View

AuthorID 1

FirstName Paul

LastName Deitel

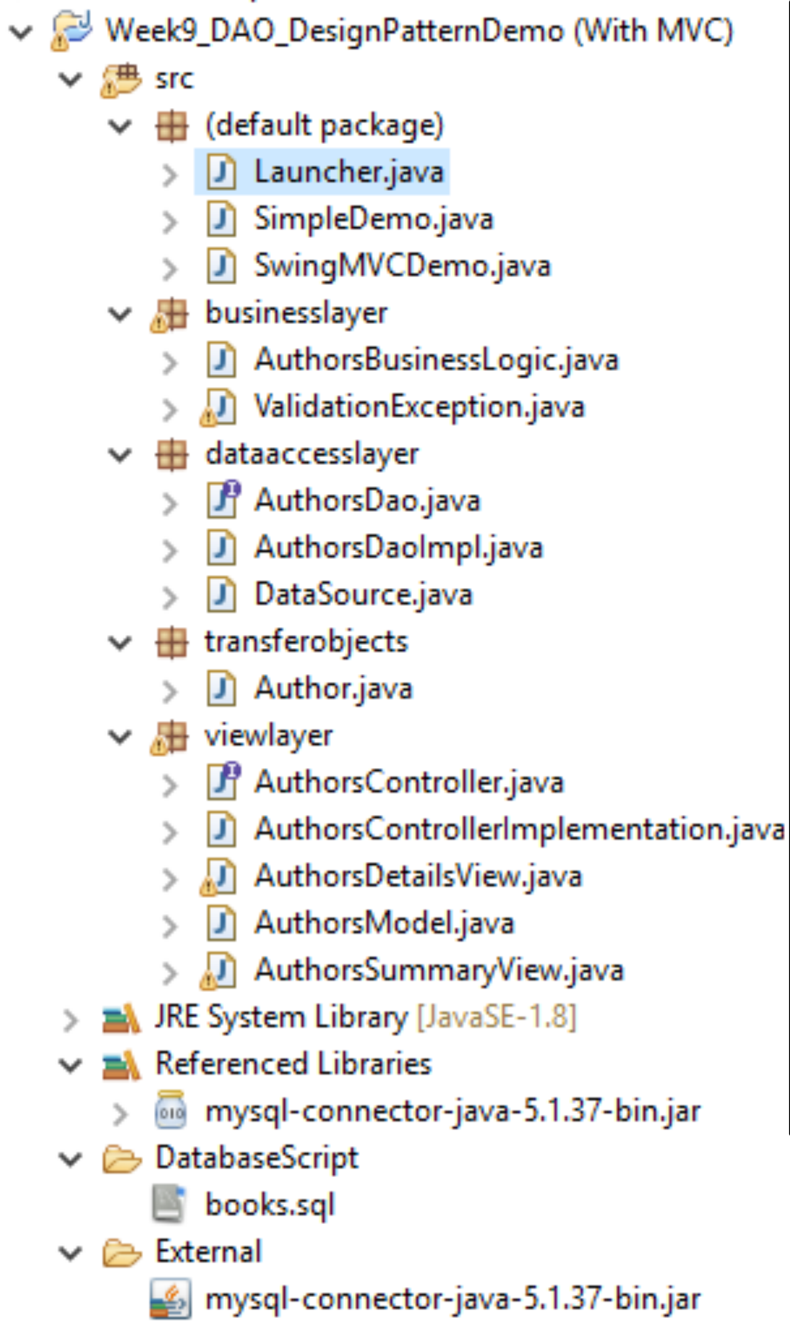

Title Summary View

AuthorID	First Name	Last Name
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
4	Michael	Morgano
5	Eric	Kern

Figure 4.0 GUI program in use.

- Navigate using buttons
- Edit text then use Add or Update (Simple program has no UI States)
- Will talk about need for Refresh button in future lecture (Concurrency)
- Click a row in the summary view to jump to that record in the Details view, MVC keeps the two views synchronized based on common model.

Structure of Project



```
Week9_DAO_DesignPatternDemo (With MVC)
  src
    (default package)
      Launcher.java
      SimpleDemo.java
      SwingMVCDemo.java
    businesslayer
      AuthorsBusinessLogic.java
      ValidationException.java
    dataaccesslayer
      AuthorsDao.java
      AuthorsDaoImpl.java
      DataSource.java
    transferobjects
      Author.java
    viewlayer
      AuthorsController.java
      AuthorsControllerImplementation.java
      AuthorsDetailsView.java
      AuthorsModel.java
      AuthorsSummaryView.java
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
    mysql-connector-java-5.1.37-bin.jar
  DatabaseScript
    books.sql
  External
    mysql-connector-java-5.1.37-bin.jar
```

Packages were used to organize the code into data access layer, business layer, and view layer.

The business layer + data access layer + data transfer objects packages are the implementation of the DAO design pattern.

To help organize the code the data transfer object (DTO) was moved into a separate package, the view will need to use the DTO but the view should not use the data access layer directly (i.e. view should not import dataaccesslayer package).

The view is a variation on the MVC from a previous lecture.

Note: For a Tiered Approach I would use Separate Projects for each Tier.

Figure 5.0 Project structure in Eclipse

Debugging tips

- Common Problems
  - Is the database running?
  - Are the connection-string, username, and password correct?
  - Was the database schema created in the database engine e.g. 'books'
  - Is there actually data in the database tables?
  - Mistakes in SQL?
  - Mistakes in PreparedStatement parameters?
  - Copy and paste mistakes in multiple layers where the data flows?
  - Validation mistakes / problems?
  - GUI logic / state problems? (Are the event handlers registered?)

JDBC Note!

- Note that numbers for SQL parameters e.g. (?, ?, ?) 1, 2, 3 might not actually work will depend on order of method calls in code top to bottom…

```
pstmt = con.prepareStatement("UPDATE Authors SET FirstName = ?, LastName = ? WHERE AuthorID = ?");
pstmt.setString(1, author.getFirstName()); // used for 1st ? in the SQL
pstmt.setString(2, author.getLastName());  // used for 2nd ? in the SQL
pstmt.setInt(3, author.getAuthorID().intValue()); // used for 3rd ? in the SQL
```

Caution!: Some JDBC drivers, jdbc-odbc-bridge, (possibly more) ignore the numbers for the parameters and simply assign each one top-down as they occur in sequence in the code.

i.e. this will result in run-time and / or logic errors:

```
pstmt = con.prepareStatement("UPDATE Authors SET FirstName = ?, LastName = ? WHERE AuthorID = ?");
pstmt.setString(2, author.getLastName());  // has 2, but will be used for 1st ?
pstmt.setInt(3, author.getAuthorID().intValue()); // has 3, but will be used for 2nd ?
pstmt.setString(1, author.getFirstName()); // has 1, but will be used for 3rd ?
```

Required Readings

Oracle. (n.d.). Core J2EE Patterns - Data Access Object. [web page] Retrieved from http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

Ram N. (2013a). Data Access Object Design Pattern - Introduction. [video] Retrieved from https://www.youtube.com/watch?v=9fVQ_mvzV48
(3 minutes 40 seconds)

Ram N. (2013b). Data Access Object Design Pattern - Class and Sequence Diagram. [video] Retrieved from https://www.youtube.com/watch?v=1ui5yVMivTo
(6 minutes 41 seconds)

Ram N. (2013c). Data Access Object Design Pattern - Implementation. [video] Retrieved from https://www.youtube.com/watch?v=H1mePFyqqiE
(14 minutes 50 seconds)

Ram N. (2013d). Data Access Object Design Pattern - Key Points. [webpage] Retrieved from http://ramj2ee.blogspot.in/2013/10/data-access-object-pattern-key-points.html

Source code for the 3rd video is available from the author's Blog, scroll down the page at: http://ramj2ee.blogspot.in/search?q=dao

Recommended Readings

Deepak Alur, John Crupi, Dan Malks. (Dec 12, 2009). Core J2EE Patterns: Data Access Object Pattern. [webpage] Retrieved from
http://www.informit.com/articles/article.aspx?p=1398621&seqNum=3

Oracle. (2001-2002). Core J2EE Patterns - Transfer Object. [web page] Retrieved from http://www.oracle.com/technetwork/java/transferobject-139757.html

Oracle. (2015). The Java Tutorials: How to Use Tables. Retrieved from http://docs.oracle.com/javase/tutorial/uiswing/components/table.html

Martin Fowler. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional. (This book is a very good reference for your future career.)
Visit Algonquin College Library in Safari Books http://www.algonquincollege.com/library/home2/
(Green Rectangle Link "Books & Videos" → Grey Tab "Ebooks" → Safari → Use "Go to Safari Ebooks")
Part 1: Chapter 1: Layering →The Three Principle Layers
Part 2: Chapter 15: Distribution Patterns →Data Transfer Object
Part 2: Chapter 18: Base Patterns → Value Object

Gerd Wagner. (June 28, 2012). Model-Driven Development of Web Applications. [webpage] Retrieved from
https://oxygen.informatik.tu-cottbus.de/IT/Lehre/MDD-Tutorial/#d5e126
(How to model database tables using UML Class diagrams)

Niraj Bhatt. (July 15, 2011). Association vs. Dependency vs. Aggregation vs. Composition. Retrieved from https://nirajrules.wordpress.com/2011/07/15/association-vs-dependency-vs-aggregation-vs-composition/

Additional Sources Cited

Deitel and Deitel. (2015). Java How to Program, 10th Edition, Pearson Education Inc.
ISBN: 978-0-13-380780-6

Appendix: Source Code for Project

```
package transferobjects;
public class Author {
    private Integer authorID;
    private String firstName;
    private String lastName;
    public Integer getAuthorID(){
        return authorID;
    }
    public void setAuthorID(Integer authorID){
        this.authorID = authorID;
    }
    public String getFirstName(){
        return firstName;
    }
    public void setFirstName(String firstName){
        this.firstName = firstName;
    }
    public String getLastName(){
        return lastName;
    }
    public void setLastName(String lastName){
        this.lastName = lastName;
    }
}

/* References:
 * Ram N. (2013).  Data Access Object Design Pattern or DAO Pattern [blog] Retrieved from
 * http://ramj2ee.blogspot.in/2013/08/data-access-object-design-pattern-or.html
 */
package dataaccesslayer;
import java.util.List;
import transferobjects.Author;
public interface AuthorsDao {
    List<Author> getAllAuthors();
    //List<Author> getAuthorsByFirstName(String firstName);
    Author getAuthorByAuthorId(Integer authorID);
    void addAuthor(Author author);
    void updateAuthor(Author author);
    void deleteAuthor(Author author);
}

/* References:
 * Ram N. (2013).  Data Access Object Design Pattern or DAO Pattern [blog] Retrieved from
 * http://ramj2ee.blogspot.in/2013/08/data-access-object-design-pattern-or.html
 */
package dataaccesslayer;
import java.util.List;
import transferobjects.Author;
import java.util.ArrayList;
import java.sql.PreparedStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
public class AuthorsDaoImpl implements AuthorsDao{
    @Override
    public List<Author> getAllAuthors() {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        ArrayList<Author> authors = null;
        try{
            DataSource ds = new DataSource();
            con = ds.createConnection();
            pstmt = con.prepareStatement(
                    "SELECT AuthorID, FirstName, LastName FROM Authors ORDER BY AuthorID");
            rs = pstmt.executeQuery();
            authors = new ArrayList<Author>();
            while(rs.next()){
                Author author = new Author();
                author.setAuthorID(new Integer(rs.getInt("AuthorID")));
                author.setFirstName(rs.getString("FirstName"));
                author.setLastName(rs.getString("LastName"));
                authors.add(author);
            }
        }
        catch(SQLException e){
            e.printStackTrace();
        }
        finally{
            try{ if(rs != null){ rs.close(); } }
            catch(SQLException ex){System.out.println(ex.getMessage());}
            try{ if(pstmt != null){ pstmt.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
            try{ if(con != null){ con.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
        }
        return authors;
    }
    @Override
    public Author getAuthorByAuthorId(Integer authorID) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        Author author = null;
        try{
            DataSource ds = new DataSource();
            con = ds.createConnection();
            pstmt = con.prepareStatement(
                    "SELECT AuthorID, FirstName, LastName FROM Authors WHERE AuthorID = ?");
            pstmt.setInt(1, authorID.intValue());
            rs = pstmt.executeQuery();
            while(rs.next()){
                author = new Author();
                author.setAuthorID(new Integer(rs.getInt("AuthorID")));
                author.setFirstName(rs.getString("FirstName"));
                author.setLastName(rs.getString("LastName"));
            }
        }
        catch(SQLException e){
            e.printStackTrace();
        }
        finally{
            try{ if(rs != null){ rs.close(); } }
            catch(SQLException ex){System.out.println(ex.getMessage());}
            try{ if(pstmt != null){ pstmt.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
            try{ if(con != null){ con.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
        }
        return author;
    }
    @Override
    public void addAuthor(Author author) {
        Connection con = null;
        PreparedStatement pstmt = null;
        try{
            DataSource ds = new DataSource();
            con = ds.createConnection();
            // do not insert AuthorID, it is generated by Database
            pstmt = con.prepareStatement(
                    "INSERT INTO Authors (FirstName, LastName) " +
                    "VALUES(?, ?)");
            pstmt.setString(1, author.getFirstName());
            pstmt.setString(2, author.getLastName());
            pstmt.executeUpdate();
        }
        catch(SQLException e){
            e.printStackTrace();
        }
        finally{
            try{ if(pstmt != null){ pstmt.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
            try{ if(con != null){ con.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
        }
    }
    @Override
    public void updateAuthor(Author author) {
            Connection con = null;
            PreparedStatement pstmt = null;
            try{
                DataSource ds = new DataSource();
                con = ds.createConnection();
                pstmt = con.prepareStatement(
                        "UPDATE Authors SET FirstName = ?, " +
                        "LastName = ? WHERE AuthorID = ?");
                pstmt.setString(1, author.getFirstName());
                pstmt.setString(2, author.getLastName());
                pstmt.setInt(3, author.getAuthorID().intValue());
                pstmt.executeUpdate();
            }
            catch(SQLException e){
                e.printStackTrace();
            }
            finally{
                try{ if(pstmt != null){ pstmt.close(); }}
                catch(SQLException ex){System.out.println(ex.getMessage());}
                try{ if(con != null){ con.close(); }}
                catch(SQLException ex){System.out.println(ex.getMessage());}
            }
    }
    @Override
    public void deleteAuthor(Author author) {
        Connection con = null;
        PreparedStatement pstmt = null;
        try{
            DataSource ds = new DataSource();
            con = ds.createConnection();
            pstmt = con.prepareStatement(
                    "DELETE FROM Authors WHERE AuthorID = ?");
            pstmt.setInt(1, author.getAuthorID().intValue());
            pstmt.executeUpdate();
        }
        catch(SQLException e){
            e.printStackTrace();
        }
        finally{
            try{ if(pstmt != null){ pstmt.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
            try{ if(con != null){ con.close(); }}
            catch(SQLException ex){System.out.println(ex.getMessage());}
        }
    }
}

/* References:
 * Ram N. (2013).  Data Access Object Design Pattern or DAO Pattern [blog] Retrieved from
 * http://ramj2ee.blogspot.in/2013/08/data-access-object-design-pattern-or.html
 */
package dataaccesslayer;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DataSource {
    private Connection connection = null;
    private final String connectionString = "jdbc:mysql://localhost/books";
    private final String username = "scott";
    private final String password = "tiger";
    public DataSource(){}
    /* Only use one connection for this application, prevent memory leaks.
     */
    public Connection createConnection(){
        try{
            if(connection != null){
                System.out.println("Cannot create new connection, one exists already");
            }
            else{
                connection = DriverManager.getConnection(connectionString, username, password);
            }
        }
        catch(SQLException ex){
            ex.printStackTrace();
        }
        return connection;
    }
}

/* References:
 * Ram N. (2013).  Data Access Object Design Pattern or DAO Pattern [blog] Retrieved from
 * http://ramj2ee.blogspot.in/2013/08/data-access-object-design-pattern-or.html
 */
package businesslayer;
import java.util.List;
import dataaccesslayer.AuthorsDao;
import dataaccesslayer.AuthorsDaoImpl;
import transferobjects.Author;

public class AuthorsBusinessLogic {
    private static final int FIRST_NAME_MAX_LENGTH = 30;
    private static final int LAST_NAME_MAX_LENGTH = 30;

    private AuthorsDao authorsDao = null;

    public AuthorsBusinessLogic(){
        authorsDao = new AuthorsDaoImpl();
    }

    public List<Author> getAllAuthors(){
        return authorsDao.getAllAuthors();
    }

    public Author getAuthor(Integer authorID){
        return authorsDao.getAuthorByAuthorId(authorID);
    }

    public void addAuthor(Author author) throws ValidationException{
        cleanAuthor(author);
        validateAuthor(author);
        authorsDao.addAuthor(author);
    }

    public void updateAuthor(Author author) throws ValidationException{
        cleanAuthor(author);
        validateAuthor(author);
        authorsDao.updateAuthor(author);
    }

    public void deleteAuthor(Author author){
        authorsDao.deleteAuthor(author);
    }

    private void cleanAuthor(Author author){
        if(author.getFirstName() != null){
            author.setFirstName(author.getFirstName().trim());
        }
        if(author.getLastName() != null){
            author.setLastName(author.getLastName().trim());
        }
    }

    private void validateAuthor(Author author) throws ValidationException{
        validateString(author.getFirstName(), "First Name", FIRST_NAME_MAX_LENGTH, true);
        validateString(author.getLastName(), "Last Name", LAST_NAME_MAX_LENGTH, true);
    }

    private void validateString(String value, String fieldName, int maxLength, boolean
isNullAllowed)
        throws ValidationException{
        if(value == null && isNullAllowed){
            // return; // null permitted, nothing to validate
        }
        else if(value == null && ! isNullAllowed){
            throw new ValidationException(String.format("%s cannot be null",
                    fieldName));
        }
        else if(value.length() == 0){
            throw new ValidationException(String.format("%s cannot be empty or only whitespace",
                    fieldName));
        }
        else if(value.length() > maxLength){
            throw new ValidationException(String.format("%s cannot exceed %d characters",
                    fieldName, maxLength));
        }
    }
    /*
    private void validateInt(int value, String fieldName)
        throws ValidationException{
        if(value <= 0){
            throw new ValidationException(String.format("%s cannot be a negative number",
                    fieldName));
        }
    }
    */
}

package businesslayer;
public class ValidationException extends Exception {
    public ValidationException(){
        super("Data not in valid format");
    }
    public ValidationException(String message){
        super(message);
    }
    public ValidationException(String message, Throwable throwable){
        super(message, throwable);
    }
    public ValidationException(Throwable throwable){
        super(throwable);
    }
}

package viewlayer;
import businesslayer.AuthorsBusinessLogic;
import businesslayer.ValidationException;
import transferobjects.Author;
import java.util.List;
import java.util.ArrayList;
import java.util.Observable;

public class AuthorsModel extends Observable{
    private int recordIndex = -1;
    private AuthorsBusinessLogic businessLogic;
    private List<Author> authors;

    public AuthorsModel(){
        businessLogic = new AuthorsBusinessLogic();
        authors = businessLogic.getAllAuthors();
    }

    public void firstAuthor(){
        if(authors != null && authors.size() > 0){
            recordIndex = 0;
            setChanged();
            notifyObservers(new Integer(recordIndex));
        }
    }

    public void previousAuthor(){
        if(authors != null && authors.size() > 0){
            if(recordIndex - 1 >= 0){
                recordIndex--;
                setChanged();
                notifyObservers(new Integer(recordIndex));
            }
        }
    }

    public void nextAuthor(){
        if(authors != null && authors.size() > 0){
            if(recordIndex + 1 < authors.size()){
                recordIndex++;
                setChanged();
                notifyObservers(new Integer(recordIndex));
            }
        }
    }

    public void lastAuthor(){
        if(authors != null && authors.size() > 0){
            recordIndex = authors.size() - 1;
            setChanged();
            notifyObservers(new Integer(recordIndex));
        }
    }

    public int getRecordIndex(){
        return recordIndex;
    }

    public Author getCurrentAuthor(){
        return authors.get(recordIndex);
    }

    public List<Author> getAllAuthors(){
        return new ArrayList<Author>(authors); // provide shallow copy
    }
    public void create(Author author) throws ValidationException{
        businessLogic.addAuthor(author);
        authors = businessLogic.getAllAuthors();
        firstAuthor();
    }

    public void readAllAuthors(){ // refreshes from database
        authors = businessLogic.getAllAuthors();
        firstAuthor();
    }

    public void update(Author author) throws ValidationException{
        businessLogic.updateAuthor(author);
        authors = businessLogic.getAllAuthors();
        firstAuthor();
    }

    public void delete(Author author){
        businessLogic.deleteAuthor(author);
        authors = businessLogic.getAllAuthors();
        firstAuthor();
    }

    public void selectByAuthorID(Integer authorID){
        if(authors != null && authors.size() > 0){
            int foundAt = -1;
            for(int index = 0; index < authors.size(); index++){
                if(authors.get(index).getAuthorID().equals(authorID)){
                    foundAt = index;
                    break;
                }
            }
            if(foundAt != -1){
                recordIndex = foundAt;
                setChanged();
                notifyObservers(new Integer(recordIndex));
            }
        }
    }
}

package viewlayer;
import java.util.Observer;
import businesslayer.ValidationException;
import transferobjects.Author;
import java.sql.SQLException;
public interface AuthorsController {
    void addObserver(Observer o);
    void first();
    void previous();
    void next();
    void last();
    void create(Author author) throws ValidationException, SQLException;
    void readAllAuthors() throws SQLException;
    void update(Author author) throws ValidationException, SQLException;
    void delete(Author author) throws SQLException;
    void selectByAuthorID(Integer authorID);
}

package viewlayer;
import java.util.Observer;
```

```java
import businesslayer.ValidationException;
import transferobjects.Author;
import java.sql.SQLException;
public class AuthorsControllerImplementation implements AuthorsController{

    private final AuthorsModel model = new AuthorsModel();

    @Override
    public void addObserver(Observer o) {
        model.addObserver(o);
    }

    @Override
    public void first() {
        model.firstAuthor();
    }

    @Override
    public void previous() {
        model.previousAuthor();
    }

    @Override
    public void next() {
        model.nextAuthor();
    }

    @Override
    public void last() {
        model.lastAuthor();
    }

    @Override
    public void create(Author author) throws ValidationException, SQLException{
        model.create(author);
    }

    @Override
    public void readAllAuthors() throws SQLException{
        model.readAllAuthors();
    }

    @Override
    public void update(Author author) throws ValidationException, SQLException{
        model.update(author);
    }

    @Override
    public void delete(Author author) throws SQLException{
        model.delete(author);
    }

    @Override
    public void selectByAuthorID(Integer authorID){
        model.selectByAuthorID(authorID);
    }
}

package viewlayer;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;
import transferobjects.Author;
import javax.swing.JButton;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Observable;
import java.util.Observer;

public class AuthorsDetailsView extends JFrame implements Observer{

    private AuthorsController controller = null;

    //ISBN, Title, EditionNumber, Copyright
    private JTextField authorIDJTextField = new JTextField();
    private JTextField firstNameJTextField = new JTextField();
    private JTextField lastNameJTextField = new JTextField();
    private JButton firstJButton = new JButton("First");
    private JButton previousJButton = new JButton("Previous");
    private JButton nextJButton = new JButton("Next");
    private JButton lastJButton = new JButton("Last");
    private JButton addJButton = new JButton("Add");
    private JButton updateJButton = new JButton("Update");
    private JButton deleteJButton = new JButton("Delete");
    private JButton refreshJButton = new JButton("Refresh");
    private JLabel recordIndicatorJLabel = new JLabel();

    public AuthorsDetailsView(AuthorsController controller){
        super("Authors Detail View");
        this.controller = controller;
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel labels = buildLabelPanel();
        JPanel texts = buildTextPanel();
        JPanel navigation = buildNavigationPanel();
        JPanel manipulation = buildManipulationPanel();
        registerEvents();
        this.add(labels, BorderLayout.WEST);
        this.add(texts, BorderLayout.CENTER);
        JPanel southPanel = new JPanel(new GridLayout(2,1));
        southPanel.add(navigation);
        southPanel.add(manipulation);
        this.add(southPanel, BorderLayout.SOUTH);
        this.pack();
        this.setVisible(true);
    }

    @Override
    public void update(Observable observable, Object object) {
        if(observable instanceof AuthorsModel && object instanceof Integer){
            AuthorsModel model = (AuthorsModel)observable;
            Integer index = (Integer)object;
            Author author = model.getCurrentAuthor();
            authorIDJTextField.setText(author.getAuthorID() + "");
            firstNameJTextField.setText(author.getFirstName());
            lastNameJTextField.setText(author.getLastName() + "");
            recordIndicatorJLabel.setText((index + 1) + " of " + model.getAllAuthors().size());
        }
    }
    private JPanel buildLabelPanel(){
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3,1));
        panel.add(new JLabel("AuthorID"));
        panel.add(new JLabel("FirstName"));
        panel.add(new JLabel("LastName"));
        return panel;
    }

    private JPanel buildTextPanel(){
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 1));
        panel.add(authorIDJTextField);
        panel.add(firstNameJTextField);
        panel.add(lastNameJTextField);
        authorIDJTextField.setEditable(false); // cannot change author id
        return panel;
    }

    private JPanel buildNavigationPanel(){
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 5));
        panel.add(firstJButton);
        panel.add(previousJButton);
        panel.add(recordIndicatorJLabel);
        panel.add(nextJButton);
        panel.add(lastJButton);
        return panel;
    }

    private JPanel buildManipulationPanel(){
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 5));
        panel.add(new JLabel()); // spacer
        panel.add(addJButton);
        panel.add(updateJButton);
        panel.add(deleteJButton);
        panel.add(refreshJButton);
        return panel;
    }

    private void registerEvents(){
        addJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    Author author = new Author();
                    author.setAuthorID(Integer.valueOf(authorIDJTextField.getText()));
                    author.setFirstName(firstNameJTextField.getText());
                    author.setLastName(lastNameJTextField.getText());
                    controller.create(author);
                }
                catch(NumberFormatException ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Enter only numbers for edition number",
                            "input error", JOptionPane.ERROR_MESSAGE);
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        updateJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    Author author = new Author();
                    author.setAuthorID(Integer.valueOf(authorIDJTextField.getText()));
                    author.setFirstName(firstNameJTextField.getText());
                    author.setLastName(lastNameJTextField.getText());
                    controller.update(author);
                }
                catch(NumberFormatException ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Enter only numbers for edition number",
                            "input error", JOptionPane.ERROR_MESSAGE);
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        deleteJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    Author author = new Author();
                    author.setAuthorID(Integer.valueOf(authorIDJTextField.getText()));
                    controller.delete(author);
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        refreshJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    controller.readAllAuthors();
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        firstJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    controller.first();
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        previousJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    controller.previous();
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        nextJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    controller.next();
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        lastJButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                try{
                    controller.last();
                }
                catch(Exception ex){
                    JOptionPane.showMessageDialog(AuthorsDetailsView.this,
                            "Exception:\n" + ex.getMessage(),
                            "Problem", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
    }
}

/* File: AuthorSummaryView.java
 * Author: Stanley Pieda
 * Date: 2015
 * Description: Demonstration of DAO Design Pattern, MVC Design Pattern
 *
 *
 * References:
 * The following references were used to instantiate and manipulate the JTable, as
 * well as work around some display bugs.
 *
 * Oracle. (2015). The Java Tutorials: How to Use Tables. Retrieved from
 *     http://docs.oracle.com/javase/tutorial/uiswing/components/table.html
 *
 * Aitor Gonzalez, Branislav Lazic, Reimeus, trashgod, ProgramFOX, user3289859.
 *     (2012-2014). JScrollpane only visible after maximising window Java Swing.
 *     [online forum] Retrieved from
 *     http://stackoverflow.com/questions/12223564/jscrollpane-only-visible-after-maximising-
window-java-swing
 *
 * user236501, Peter Lang, RustyTheBoyRobot, Daniel De León, Alexey, Achille,
 *     Tim Cooper, tom, Sumit Singh, nkvnkv. (2010-2011). JTable How to refresh
 *     table model after insert delete or update the data. [online forum] Retreived from
 *     http://stackoverflow.com/questions/3179136/jtable-how-to-refresh-table-model-after-insert-
delete-or-update-the-data
 *
 * newbee, Bitmap, Fortega, Reverend Gonzo, Bharathiraja, camickr. (2010-2014). How to set header
for JTable?.
 *     [online forum] Retrieved from
 *     http://stackoverflow.com/questions/2297991/how-to-set-header-for-jtable
 */
package viewlayer;
import javax.swing.JFrame;
import javax.swing.JTable;
import javax.swing.JScrollPane;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
import transferobjects.Author;
import java.awt.BorderLayout;
import java.util.Observable;
import java.util.Observer;
import java.util.List;
public class AuthorsSummaryView extends JFrame implements Observer{
    private AuthorsController controller = null;
    private JTable table = null;
    private JScrollPane scrollPane = null;

    public AuthorsSummaryView(AuthorsController controller){
        super("Title Summary View");
        this.controller = controller;
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        String[] names = {"AuthorID", "First Name", "Last Name"};
        DefaultTableModel model = new DefaultTableModel(names, 0);
        table = new JTable(model);
        scrollPane = new JScrollPane(table);
        this.getContentPane().add(scrollPane, BorderLayout.CENTER);

        table.getSelectionModel().addListSelectionListener(new ListSelectionListener(){
            @Override
            public void valueChanged(ListSelectionEvent event){
                int rowIndex = table.getSelectedRow();
                if(rowIndex != -1){
                    Integer authorID = Integer.valueOf(table.getValueAt(rowIndex, 0).toString());
                    controller.selectByAuthorID(authorID);
                }
            }
        });
        this.setSize(400, 200);
        this.setVisible(true);;
    }

    @Override
    public void update(Observable observable, Object index) {
```

```java
        if(observable instanceof AuthorsModel && index instanceof Integer){
            List<Author> list = ((AuthorsModel)observable).getAllAuthors();
            Object o = table.getModel();
            DefaultTableModel tableModel = (DefaultTableModel) table.getModel();
            tableModel.setRowCount(0); // discards all of the old rows
            for(int i = 0; i < list.size(); i++){
                String[] row = new String[3];
                row[0] = list.get(i).getAuthorID().toString();
                row[1] = list.get(i).getFirstName();
                row[2] = list.get(i).getLastName();
                tableModel.addRow(row);
            }
            table.setModel(tableModel);
            table.repaint();
            this.repaint();
            this.revalidate();
        }
    }
}

import businesslayer.AuthorsBusinessLogic;
import businesslayer.ValidationException;
import transferobjects.Author;
import java.util.List;
public class SimpleDemo {

    public void demo(){
        try{
            AuthorsBusinessLogic logic = new AuthorsBusinessLogic();
            List<Author> list = null;
            Author author = null;

            System.out.println("Printing Authors");
            list = logic.getAllAuthors();
            printAuthors(list);

            System.out.println("Printing One Author");
            author = logic.getAuthor( new Integer(1) );
            printAuthor(author);
            System.out.println();

            System.out.println("Inserting One Author");
            author = new Author();
            author.setFirstName("FirstTestAdd");
            author.setLastName("LastTestAdd");
            logic.addAuthor(author);
            list = logic.getAllAuthors();
            printAuthors(list);

            System.out.println("Updateing last author");
            Integer updatePrimaryKey = list.get(list.size() - 1).getAuthorID();
            author = new Author();
            author.setAuthorID(updatePrimaryKey);
            author.setFirstName("FirstTestUpdate");
            author.setLastName("LastTestUpdate");
            logic.updateAuthor(author);
            list = logic.getAllAuthors();
            printAuthors(list);

            System.out.println("Deleteing last author");
            author = list.get(list.size() - 1);
            logic.deleteAuthor(author);
            list = logic.getAllAuthors();
            printAuthors(list);
        }
        catch(ValidationException e){
            System.err.println(e.getMessage());
        }

    }

    private static void printAuthor(Author author){
            String output = String.format("%s, %s, %s",
                    author.getAuthorID().toString(),
                    author.getFirstName(),
                    author.getLastName());
            System.out.println(output);
    }

    private static void printAuthors(List<Author> authors){
        for(Author author : authors){
            printAuthor(author);
        }
        System.out.println();
    }
}

import viewlayer.AuthorsController;
import viewlayer.AuthorsControllerImplementation;
import viewlayer.AuthorsDetailsView;
import viewlayer.AuthorsSummaryView;
public class SwingMVCDemo {
    public void demo(){
        AuthorsController controller = new AuthorsControllerImplementation();
        AuthorsDetailsView detailsView = new AuthorsDetailsView(controller);
        AuthorsSummaryView summaryView = new AuthorsSummaryView(controller);
        controller.addObserver(detailsView);
        controller.addObserver(summaryView);
    }
}

public class Launcher {
    public static void main(String[] args) {
        (new SimpleDemo()).demo();
        // (new SwingMVCDemo()).demo();
    }
}
```