Appendix B – Code

Start Menu Code:

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.Font;
import javax.swing.SwingConstants;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class StartMenu extends JFrame
{

	private JPanel contentPane;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
				{
					StartMenu frame = new StartMenu();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public StartMenu()
	{
```

```
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          setBounds(100, 100, 500, 500);
          contentPane = new JPanel();
          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
          setContentPane(contentPane);
          contentPane.setLayout(null);

          JLabel startLabel = new JLabel("LEGO Collection
Database");
          startLabel.setFont(new Font("Tahoma", Font.PLAIN,
20));

     startLabel.setHorizontalAlignment(SwingConstants.CENTER);
          startLabel.setBounds(54, 34, 365, 50);
          contentPane.add(startLabel);

          //Takes user to Log In window, closes current window
          JButton logInBttn = new JButton("Log In");
          logInBttn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    LogIn log = new LogIn();
                    setVisible(false);
                    log.setVisible(true);
               }
          });
          logInBttn.setFont(new Font("Tahoma", Font.PLAIN, 14));
          logInBttn.setBounds(122, 166, 217, 50);
          contentPane.add(logInBttn);

          //Takes user to Create Account Window
          JButton createAccountBttn = new JButton("Create
Account");
          createAccountBttn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    CreateAccount create = new CreateAccount();
                    setVisible(false);
                    create.setVisible(true);
               }
          });
          createAccountBttn.setFont(new Font("Tahoma",
Font.PLAIN, 14));
```

```
            createAccountBttn.setBounds(122, 259, 217, 50);
            contentPane.add(createAccountBttn);
        }
    }
```

Log In Code

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JButton;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.Color;

public class LogIn extends JFrame {

	private JPanel contentPane;
	private JTextField usernameField;
	private JPasswordField passwordField;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
				{
					LogIn frame = new LogIn();
					frame.setVisible(true);
				}
				catch (Exception e)
				{
					e.printStackTrace();
				}
			}
		});
	}
```

```
     /**
      * Create the frame.
      */
     public LogIn()
     {

          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          setBounds(100, 100, 500, 500);
          contentPane = new JPanel();
          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
          setContentPane(contentPane);
          contentPane.setLayout(null);

          JLabel logInLbl = new JLabel("Log In");
          logInLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));

     logInLbl.setHorizontalAlignment(SwingConstants.CENTER);
          logInLbl.setBounds(80, 26, 298, 55);
          contentPane.add(logInLbl);

          //Displays the correct errors when appropriate
          JLabel errorLbl = new JLabel("Password is
incorrect.");

     errorLbl.setHorizontalAlignment(SwingConstants.CENTER);
          errorLbl.setForeground(Color.RED);
          errorLbl.setFont(new Font("Tahoma", Font.PLAIN, 14));
          errorLbl.setBounds(64, 76, 343, 40);
          contentPane.add(errorLbl);
          errorLbl.setVisible(false);

          //Where username is entered
          JLabel usernameLbl = new JLabel("Username: ");
          usernameLbl.setFont(new Font("Tahoma", Font.PLAIN,
14));
          usernameLbl.setBounds(64, 113, 72, 31);
          contentPane.add(usernameLbl);

          usernameField = new JTextField();
          usernameField.setFont(new Font("Tahoma", Font.PLAIN,
14));
          usernameField.setBounds(64, 154, 343, 35);
          contentPane.add(usernameField);
          usernameField.setColumns(10);

          //Where password is entered
          JLabel passwordLbl = new JLabel("Password:");
```

```
          passwordLbl.setFont(new Font("Tahoma", Font.PLAIN,
14));
          passwordLbl.setBounds(64, 221, 69, 31);
          contentPane.add(passwordLbl);

          passwordField = new JPasswordField();
          passwordField.setFont(new Font("Tahoma", Font.PLAIN,
14));
          passwordField.setBounds(64, 262, 343, 35);
          contentPane.add(passwordField);

          //Closes Log In window
          //Takes user back to the Start Menu
          JButton cancelBttn = new JButton("Cancel");
          cancelBttn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    StartMenu startMenu = new StartMenu();
                    startMenu.setVisible(true);
                    setVisible(false);
               }
          });
          cancelBttn.setFont(new Font("Tahoma", Font.PLAIN,
14));
          cancelBttn.setBounds(262, 327, 145, 55);
          contentPane.add(cancelBttn);

          //If the entered username and password are correct
          //Logs user in, closes Log In window, opens View
Window
          //Otherwise displays appropriate error
          JButton LogInBtn = new JButton("Log In");
          LogInBtn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    //Error for when the entered username or
password is null
                    if
(usernameField.getText().contentEquals("") ||
passwordField.getText().contentEquals(""))
                    {
                         errorLbl.setVisible(true);
```

```
                         errorLbl.setText("Please fill in all
fields.");
                    }
                    //Checks user's log in credentials
                    else
                    {
                         Users cur = new Users(0, null, null,
usernameField.getText(), passwordField.getText());
                         Users checkUser =
cur.getUser(usernameField.getText(), passwordField.getText());
                         if (checkUser == null)
                         {
                              errorLbl.setVisible(true);
                              errorLbl.setText("Username or
password is incorrect.");
                         }
                         //Logs user in, closes Log In window,
opens View Window
                         //Otherwise displays appropriate error
                         else
                         {
                              ViewWindow view = new
ViewWindow();
                              setVisible(false);
                              view.setVisible(true);
                              view.run(checkUser.getUserId(),
checkUser.getFirstName(), checkUser.getLastName(),
                                        checkUser.getUsername(),
checkUser.getPassword());
                         }
                    }
               }
          });
          LogInBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
          LogInBtn.setBounds(64, 327, 145, 55);
          contentPane.add(LogInBtn);

     }
}
```

Create Account Code

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.Color;

public class CreateAccount extends JFrame
{

     private JPanel contentPane;
     private JTextField fNameField;
     private JTextField lastNameField;
     private JTextField usernameField;
     private JPasswordField passwordField;

     /**
      * Launch the application.
      */
     public static void main(String[] args)
     {
          EventQueue.invokeLater(new Runnable()
          {
               public void run()
               {
                    try
                    {
                         CreateAccount frame = new Cre-
ateAccount();
                         frame.setVisible(true);
                    }
                    catch (Exception e)
                    {
                         e.printStackTrace();
                    }
```

```
                }
            });
        }

        /**
         * Create the frame.
         */
        public CreateAccount()
        {
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(100, 100, 620, 750);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(null);

            JLabel acctCreationLabel = new JLabel("Account Crea-
tion");
            acctCreationLabel.setFont(new Font("Tahoma",
Font.PLAIN, 20));
            acctCreationLabel.setHorizontalAlignment(SwingCon-
stants.CENTER);
            acctCreationLabel.setBounds(144, 49, 293, 41);
            contentPane.add(acctCreationLabel);

            //Displays the correct errors when appropriate
            JLabel errorLbl = new JLabel("");
            errorLbl.setHorizontalAlignment(SwingConstants.CEN-
TER);
            errorLbl.setForeground(Color.RED);
            errorLbl.setFont(new Font("Tahoma", Font.PLAIN, 14));
            errorLbl.setBounds(65, 95, 458, 71);
            contentPane.add(errorLbl);
            errorLbl.setVisible(false);

            //Where the user enters their username
            JLabel fNameLabel = new JLabel("First Name:");
            fNameLabel.setFont(new Font("Tahoma", Font.PLAIN,
14));
            fNameLabel.setBounds(65, 156, 72, 31);
            contentPane.add(fNameLabel);

            fNameField = new JTextField();
            fNameField.setFont(new Font("Tahoma", Font.PLAIN,
14));
            fNameField.setColumns(10);
            fNameField.setBounds(65, 197, 444, 35);
```

```
            contentPane.add(fNameField);

            //Where the user enters their last name
            lastNameField = new JTextField();
            lastNameField.setFont(new Font("Tahoma", Font.PLAIN,
14));
            lastNameField.setColumns(10);
            lastNameField.setBounds(65, 301, 444, 35);
            contentPane.add(lastNameField);

            JLabel lastNameLabel = new JLabel("Last Name:");
            lastNameLabel.setFont(new Font("Tahoma", Font.PLAIN,
14));
            lastNameLabel.setBounds(65, 260, 72, 31);
            contentPane.add(lastNameLabel);

            //Where the user enters their desired username
            //The username cannot be a duplicate of a pre-existing
username
            JLabel usernameLbl = new JLabel("Username: ");
            usernameLbl.setFont(new Font("Tahoma", Font.PLAIN,
14));
            usernameLbl.setBounds(65, 364, 72, 31);
            contentPane.add(usernameLbl);

            usernameField = new JTextField();
            usernameField.setFont(new Font("Tahoma", Font.PLAIN,
14));
            usernameField.setColumns(10);
            usernameField.setBounds(65, 405, 444, 35);
            contentPane.add(usernameField);

            //Where the user enters their desired password
            JLabel lblEnterAPassword = new JLabel("Enter a pass-
word that is at least 8 characters.");
            lblEnterAPassword.setFont(new Font("Tahoma",
Font.PLAIN, 14));
            lblEnterAPassword.setBounds(65, 464, 288, 31);
            contentPane.add(lblEnterAPassword);

            passwordField = new JPasswordField();
            passwordField.setFont(new Font("Tahoma", Font.PLAIN,
14));
            passwordField.setColumns(10);
            passwordField.setBounds(65, 505, 444, 35);
            contentPane.add(passwordField);
```

```
          // Cancels account creation process
          //Takes the user back to the StartMenu
          //Closes account creation window
          JButton cancelBttn = new JButton("Cancel");
          cancelBttn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    StartMenu start = new StartMenu();
                    usernameField.setText("");
                    passwordField.setText("");
                    start.setVisible(true);
                    setVisible(false);
               }
          });
          cancelBttn.setFont(new Font("Tahoma", Font.PLAIN,
14));
          cancelBttn.setBounds(309, 583, 200, 55);
          contentPane.add(cancelBttn);

          // If conditions are fulfilled (i.e. username has not
been used before in users
          // database, user's inputted password is >= 8 charac-
ters)
          // Creates new account for user, writing to the data-
base for Users
          // And takes user to the View Window
          JButton newAcctScreen = new JButton("Create Account");
          newAcctScreen.addMouseListener(new MouseAdapter() {
               @Override
               public void mouseClicked(MouseEvent e) {
                    if (fNameField.getText().equals("") || last-
NameField.getText().equals("")
                                   || username-
Field.getText().equals("") || password-
Field.~~getText~~().equals(""))
                    {
                         errorLbl.setVisible(true);
                         errorLbl.setText("Please fill in all
fields.");
                    }
                    else
                    {
                         //Error for when the password the user
inputs is less than 8 characters
```

```
                        if (passwordField.getText().length() <
8)
                        {
                             errorLbl.setVisible(true);
                             errorLbl.setText("Password is
shorter than 8 characters. Please enter a longer password.");
                        }
                        else
                        {
                             ViewWindow view = new ViewWin-
dow();
                             Users cur = new Users(0, null,
null, usernameField.getText(), passwordField.getText());
                             //Error for if the user's inputted
username is already taken
                             if (cur.isUsernameTaken(username-
Field.getText()))
                             {
                                  errorLbl.setVisible(true);
                                  errorLbl.setText("Username is
already taken. Please a different new username.");
                             }
                             //Creates account for the new user
using the user's input, adds it to the database
                             //And takes the user to the View
Window
                             //Closes the ACcount Creation win-
dow
                             else
                             {
                                  Users newUser = new Users(0,
fNameField.getText(), lastNameField.getText(),
                                                       username-
Field.getText(), passwordField.getText());
                                  newUser.addUser();
                                  newUser =
newUser.getUser(usernameField.getText(), password-
Field.getText());
                                  setVisible(false);
                                  view.setVisible(true);
                                  view.run(newUser.getUserId(),
newUser.getFirstName(), newUser.getLastName(),

     newUser.getUsername(), newUser.getPassword());
                             }
                        }
```

```
                    }
                }
            });
            newAcctScreen.setFont(new Font("Tahoma", Font.PLAIN,
14));
            newAcctScreen.setBounds(65, 583, 192, 55);
            contentPane.add(newAcctScreen);
        }
}
```

View Window Code

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;

import org.sqlite.SQLiteDataSource;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JButton;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JTable;
import javax.swing.JComboBox;
import javax.swing.JScrollPane;
import javax.swing.JTextPane;
import java.awt.Component;
import java.awt.Panel;
import javax.swing.JTextField;
import java.awt.Color;

public class ViewWindow extends JFrame
{

     private JPanel contentPane;
     public int userId;
     public String fName;
     public String lastName;
     public String username;
     public String password;
     public int quantity;
     public String date;
```

```
	public String price;
	public String notes;
	public String setNum;
	public int row;
	public String selectedSort;
	private JTextField filterField;
	private Database collection;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
				{
					ViewWindow frame = new ViewWindow();
					frame.setVisible(true);
				} catch (Exception e)
				{
					e.printStackTrace();
				}
			}
		});
	}

	// Main method where everything is run; Otherwise certain
variables would be
	// null,
	// causing the program to crash
	public void run(int id, String first, String last, String
use, String pas)
	{
		userId = id;
		fName = first;
		lastName = last;
		username = use;
		password = pas;
		selectedSort = "Quantity";
		collection = new Database();

		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 750, 750);
		contentPane = new JPanel();
```

```
          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
          setContentPane(contentPane);
          contentPane.setLayout(null);

          // Takes user back to the starting window
          JButton logOutBttn = new JButton("Log Out");
          logOutBttn.setBounds(508, 12, 122, 41);
          logOutBttn.addMouseListener(new MouseAdapter() {
               @Override
               public void mouseClicked(MouseEvent e) {
                    StartMenu lego = new StartMenu();
                    lego.setVisible(true);
                    setVisible(false);
               }
          });

          //Shows who's collection is being displayed
          JLabel yrCollectionLbl = new JLabel(fName + "'s
Collection");
          yrCollectionLbl.setBounds(142, 10, 374, 41);

     yrCollectionLbl.setHorizontalAlignment(SwingConstants.CENTE
R);
          yrCollectionLbl.setFont(new Font("Tahoma", Font.PLAIN,
20));
          contentPane.add(yrCollectionLbl);

          //Instructs user to select a set in the table to edit
or delete
          JLabel dirLabel = new JLabel("Select set to edit or
delete it.");

     dirLabel.setHorizontalAlignment(SwingConstants.CENTER);
          dirLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
          dirLabel.setBounds(190, 38, 307, 29);
          contentPane.add(dirLabel);

          // Error label
          // Shows error when a row is not selected to edit or
delete information
          JLabel errorLbl = new JLabel("Set not selected. Please
select a set.");
          errorLbl.setForeground(Color.RED);
          errorLbl.setBounds(135, 61, 398, 29);
          errorLbl.setFont(new Font("Tahoma", Font.BOLD, 14));

     errorLbl.setHorizontalAlignment(SwingConstants.CENTER);
```

```
          contentPane.add(errorLbl);
          errorLbl.setVisible(false);

          //Text field for entering filter words
          filterField = new JTextField();
          filterField.setFont(new Font("Tahoma", Font.PLAIN,
14));
          filterField.setBounds(98, 94, 215, 31);
          contentPane.add(filterField);
          filterField.setColumns(10);

          //Where table containing all of the user's set entries
is initialized
          //Reads from the LEGO_Collection.db file
          DefaultTableModel model = new DefaultTableModel();
          model.addColumn("Qty");
          model.addColumn("Set #");
          model.addColumn("Name");
          model.addColumn("Release");
          model.addColumn("Purchase Date");
          model.addColumn("Price");
          model.addColumn("Notes");

          //Prevents the collection table's contents from being
edited outside of the Edit Entry window
          JTable table = new JTable(model)
          {
               private static final long serialVersionUID = 1L;

               public boolean isCellEditable(int row, int
column)
               {
                    return false;
               };
          };
          table.getColumnModel().getColumn(0).setWidth(5);
          table.getColumnModel().getColumn(1).setWidth(10);
          table.getColumnModel().getColumn(2).setWidth(30);
          table.getColumnModel().getColumn(3).setWidth(10);
          table.getColumnModel().getColumn(4).setWidth(10);
          table.getColumnModel().getColumn(5).setWidth(5);
          table.getColumnModel().getColumn(6).setWidth(30);

          // Connects to the LEGO_Collection database
          // So that the user's collection information can be
gathered
```

```
          // Selects all rows that have a user_id equal to the
current user's userId
          String query = "SELECT quantity, set_num, name, year,
pur_date, price, notes FROM Collection WHERE user_id = "
                    + userId + " ORDER BY " + selectedSort + "
ASC";

          SQLiteDataSource source = collection.getDs();

          try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();)
          {

               ResultSet rs = stmt.executeQuery(query);
               while (rs.next())
               {
                    model.insertRow(model.getRowCount(),
                              new String[]
{ rs.getString("quantity"), rs.getString("set_num"),
rs.getString("name"),
                                        rs.getString("year"),
rs.getString("pur_date"), rs.getString("price"),

     rs.getString("notes") });
               }
          }
          catch (SQLException e)
          {
               e.printStackTrace();
               System.exit(0);
          }
          table.setRowSelectionAllowed(true);
          JScrollPane scrollPane = new JScrollPane(table);
          scrollPane.setBounds(98, 133, 532, 402);
          contentPane.add(scrollPane);

          //Takes in user's input from filterField
          //And uses it to filter entries in the user's
collection based on the set name
          //Reinitializes the table so that it has the filtered
entries
          JButton searchBttn = new JButton("Search");
          searchBttn.setFont(new Font("Tahoma", Font.PLAIN,
14));
          searchBttn.addMouseListener(new MouseAdapter()
          {
               @Override
```

```
                public void mouseClicked(MouseEvent e)
                {
                     String text = filterField.getText();
                     String query;
                     if (!text.contentEquals(""))
                     {
                          query = "SELECT quantity, set_num,
name, year, pur_date, price, notes FROM Collection WHERE name
LIKE '%"
                                       + text + "%' AND" + " user_id
= " + userId;
                     }

                     else
                     {
                          query = "SELECT quantity, set_num,
name, year, pur_date, price, notes FROM Collection WHERE user_id
= "
                                       + userId;
                     }

                     SQLiteDataSource source =
collection.getDs();

                     try (Connection conn =
source.getConnection(); Statement stmt =
conn.createStatement();)
                     {
                          ResultSet rs =
stmt.executeQuery(query);
                          DefaultTableModel mod =
(DefaultTableModel) table.getModel();
                          mod.setRowCount(0);
                          while (rs.next())
                          {

     model.insertRow(model.getRowCount(),
                                            new String[]
{ rs.getString("quantity"), rs.getString("set_num"),
rs.getString("name"),

     rs.getString("year"), rs.getString("pur_date"),
rs.getString("price"),

     rs.getString("notes") });
                          }
                     }
```

```
                    catch (SQLException j)
                    {
                         j.printStackTrace();
                         System.exit(0);
                    }
               }
          });
          searchBttn.setBounds(315, 94, 84, 31);
          contentPane.add(searchBttn);

          // Allows user to sort entries based on column headers
          // Entries sorted in ascending order
          JLabel sortInstructions = new JLabel("Sort by: ");
          sortInstructions.setBounds(444, 89, 62, 34);
          sortInstructions.setFont(new Font("Tahoma",
Font.PLAIN, 14));

     sortInstructions.setHorizontalAlignment(SwingConstants.LEFT
);
          contentPane.add(sortInstructions);

          JComboBox<String> sortOptions = new
JComboBox<String>();
          sortOptions.setBounds(516, 92, 114, 29);
          sortOptions.setFont(new Font("Tahoma", Font.PLAIN,
14));
          sortOptions.setEnabled(true);
          //Adds list of options
          sortOptions.addItem("Quantity");
          sortOptions.addItem("Set #");
          sortOptions.addItem("Name");
          sortOptions.addItem("Release");
          sortOptions.addItem("Date Purchased");
          sortOptions.addItem("Purchase Price");
          sortOptions.addActionListener(new ActionListener()
          {
               public void actionPerformed(ActionEvent e)
               {
                    //Gets the String selected from the
sortOptions box
                    String sort =
sortOptions.getSelectedItem().toString();

                    //Based on the contents of sort
                    //Changes selectedSort to the appropriate
column header to use in the query String
                    switch (sort)
```

```
                    {
                    case "Quantity":
                         selectedSort = "quantity";
                         break;
                    case "Set #":
                         selectedSort = "set_num";
                         break;
                    case "Name":
                         selectedSort = "name";
                         break;
                    case "Release":
                         selectedSort = "year";
                         break;
                    case "Date Purchased":
                         selectedSort = "pur_date";
                         break;
                    case "Purchase Price":
                         selectedSort = "price";
                         break;
                    }

                    // Deletes table's contents and
reinitializes them using the given specifications in
                    // the query String
                    String query = "SELECT quantity, set_num,
name, year, pur_date, price, notes FROM Collection WHERE user_id
= "
                              + userId + " ORDER BY " +
selectedSort + " ASC";

                    SQLiteDataSource source =
collection.getDs();

                    try (Connection conn =
source.getConnection(); Statement stmt =
conn.createStatement();)
                    {

                         ResultSet rs =
stmt.executeQuery(query);
                         DefaultTableModel mod =
(DefaultTableModel) table.getModel();
                         mod.setRowCount(0);
                         while (rs.next())
                         {

     model.insertRow(model.getRowCount(),
```

```
                                        new String[]
{ rs.getString("quantity"), rs.getString("set_num"),
rs.getString("name"),

     rs.getString("year"), rs.getString("pur_date"),
rs.getString("price"),

     rs.getString("notes") });
                         }
                    }
                    catch (SQLException r)
                    {
                         r.printStackTrace();
                         System.exit(0);
                    }
               }
          });
          contentPane.add(sortOptions);

          // Takes user to Add Entry window
          JButton addEntry = new JButton("Add Entry");
          addEntry.setBounds(98, 566, 170, 55);
          contentPane.add(addEntry);
          addEntry.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    AddEntry add = new AddEntry();
                    setVisible(false);
                    add.setVisible(true);
                    add.run(userId, fName, lastName, username,
password);
               }
          });
          addEntry.setFont(new Font("Tahoma", Font.PLAIN, 14));

          // Takes user to Edit Entry window if a set is
selected
          // Otherwise causes an error to pop up
          JButton editEntry = new JButton("Edit Entry");
          editEntry.setBounds(278, 566, 170, 55);
          contentPane.add(editEntry);
          editEntry.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
```

```
                {
                     int row = table.getSelectedRow();
                     if (row >= 0)
                     {
                          EditEntry editor = new EditEntry();
                          editor.setVisible(true);
                          setVisible(false);
                          quantity = Integer.parseInt((String)
table.getValueAt(row, 0));
                          date = table.getValueAt(row,
4).toString();
                          price = table.getValueAt(row,
5).toString();
                          notes = table.getValueAt(row,
6).toString();
                          editor.run(userId, fName, lastName,
username, password,

     Integer.parseInt(table.getValueAt(row, 0).toString()),
table.getValueAt(row, 1).toString(),
                                        table.getValueAt(row,
2).toString(), Integer.parseInt(table.getValueAt(row,
3).toString()),
                                        table.getValueAt(row,
4).toString(), table.getValueAt(row, 5).toString(),
                                        table.getValueAt(row,
6).toString());
                     }
                     else
                     {
                          errorLbl.setVisible(true);
                     }
                }
           });
           editEntry.setFont(new Font("Tahoma", Font.PLAIN, 14));


           //Pop up for deleting entries
           Panel deletePop = new Panel();
           deletePop.setBounds(157, 541, 418, 120);
           contentPane.add(deletePop);
           deletePop.setLayout(null);
           deletePop.setVisible(false);

           JLabel youSureLbl = new JLabel("Are you sure you want
to delete this set?");
```

```
          youSureLbl.setFont(new Font("Tahoma", Font.PLAIN,
14));

     youSureLbl.setHorizontalAlignment(SwingConstants.CENTER);
          youSureLbl.setBounds(76, 27, 261, 17);
          deletePop.add(youSureLbl);
          youSureLbl.setVisible(false);

          //Makes the pop-up window for deleting entries appear
          JButton deleteEntry = new JButton("Delete Entry");
          deleteEntry.setBounds(460, 566, 170, 55);
          deleteEntry.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    int row = table.getSelectedRow();

                    if (row < 0)
                    {
                         errorLbl.setVisible(true);
                    } else
                    {
                         deletePop.setVisible(true);
                         youSureLbl.setVisible(true);
                         addEntry.setVisible(false);
                         editEntry.setVisible(false);
                         deleteEntry.setVisible(false);
                    }

               }
          });

          if (!deletePop.isVisible())
          {
               deleteEntry.setVisible(true);
          }

          deleteEntry.setFont(new Font("Tahoma", Font.PLAIN,
14));
          contentPane.add(deleteEntry);

          // If selected, closes pop up window
          JButton noBttn = new JButton("No");
          noBttn.setFont(new Font("Tahoma", Font.PLAIN, 14));
          noBttn.setBounds(111, 64, 60, 35);
          noBttn.addMouseListener(new MouseAdapter()
```

```
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    deletePop.setVisible(false);
                    row = -1;
               }
          });
          deletePop.add(noBttn);

          // If selected, deletes set from the Collection table
in LEGO_Collection
          // database
          // And re-initializes table appropriately
          JButton yesBttn = new JButton("Yes");
          yesBttn.setFont(new Font("Tahoma", Font.PLAIN, 14));
          yesBttn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {

                    Collection cur = new Collection(userId, 0,
table.getValueAt(row, 1).toString(), null, 0, null, null,
                              null);
                    cur.deleteEntry();

                    DefaultTableModel mod = (DefaultTableModel)
table.getModel();
                    mod.setRowCount(0);
                    String query = "SELECT quantity, set_num,
name, year, pur_date, price, notes FROM Collection WHERE user_id
= "
                              + userId + "";
                    SQLiteDataSource source =
collection.getDs();
                    try (Connection conn =
source.getConnection(); Statement stmt =
conn.createStatement();)
                    {
                         ResultSet rs =
stmt.executeQuery(query);
                         while (rs.next())
                         {

     model.insertRow(model.getRowCount(),
```

```
                                        new String[]
{ rs.getString("quantity"), rs.getString("set_num"),
rs.getString("name"),

     rs.getString("year"), rs.getString("pur_date"),
rs.getString("price"),

     rs.getString("notes") });
                         }
                    }
                    catch (SQLException r)
                    {
                         r.printStackTrace();
                         System.exit(0);
                    }

                    youSureLbl.setVisible(false);
                    deletePop.setVisible(false);
                    addEntry.setVisible(true);
                    editEntry.setVisible(true);
                    deleteEntry.setVisible(true);
               }
          });
          yesBttn.setBounds(222, 64, 60, 35);
          deletePop.add(yesBttn);
          logOutBttn.setFont(new Font("Tahoma", Font.PLAIN,
14));
          contentPane.add(logOutBttn);
     }

     /**
      * Create the frame.
      */
     // Empty because otherwise certain variables would be null
     // Causing the program to crash
     public ViewWindow() {

     }
}
```

Add Entry Code

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFormattedTextField;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.NumberFormat;
import java.util.Properties;

import org.jdatepicker.impl.JDatePanelImpl;
import org.jdatepicker.impl.JDatePickerImpl;
import org.jdatepicker.impl.UtilDateModel;
import org.sqlite.SQLiteDataSource;

import javax.swing.SwingConstants;
import javax.swing.SpringLayout;

public class AddEntry extends JFrame
{

     private JPanel contentPane;
     private JTextField filterField;
     private JTextField purPrice;
```

```
	private JTextField notesField;
	public String fName;
	public String lastName;
	public int userId;
	public String username;
	public String password;
	public int quantity;
	public String setNum;
	public String date;
	public String price;
	public String notes;
	private int row;
	private String selectedSort;
	private JTextField qtyField;
	private Database catalogue;
	private SpringLayout springLayout; //uh

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
				{
					AddEntry frame = new AddEntry();
					frame.setVisible(true);
				}
				catch (Exception e)
				{
					e.printStackTrace();
				}
			}
		});
	}

	// Main method where everything is run; Otherwise certain
variables would be
	// null
	// causing the program to crash
	public void run(int id, String first, String last, String
use, String pas)
	{
		userId = id;
```

```
            fName = first;
            lastName = last;
            username = use;
            password = pas;
            catalogue = new Database();

            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(100, 100, 750, 750);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(null);

            // Directions for adding entries
            // Changes depending what part of process user is on
            JLabel AddEntryDir = new JLabel("Select set to add
it.");
            AddEntryDir.setHorizontalAlignment(SwingConstants.CEN-
TER);
            AddEntryDir.setFont(new Font("Tahoma", Font.PLAIN,
16));
            AddEntryDir.setBounds(179, 22, 336, 22);
            contentPane.add(AddEntryDir);

            //Displays the correct errors when appropriate
            JLabel errorLbl = new JLabel("");
            errorLbl.setFont(new Font("Tahoma", Font.BOLD, 10));
            errorLbl.setHorizontalAlignment(SwingConstants.CEN-
TER);
            errorLbl.setForeground(Color.RED);
            errorLbl.setBounds(57, 48, 615, 22);
            contentPane.add(errorLbl);
            errorLbl.setVisible(false);

            //Text field for entering filter words
            filterField = new JTextField();
            filterField.setFont(new Font("Tahoma", Font.PLAIN,
14));
            filterField.setBounds(98, 80, 215, 31);
            contentPane.add(filterField);
            filterField.setColumns(10);

            // Table where catalogue of all LEGO Sets ever pro-
duced
            // is displayed. Table initialized based on the
LEGO_Database table in the
            // LEGO_Collection table
```

```
          DefaultTableModel model = new DefaultTableModel();
          model.addColumn("Set #");
          model.addColumn("Name");
          model.addColumn("Year Released");
          model.addColumn("# of Parts");

          //Prevents catalogue table's contents from being ed-
ited
          JTable table = new JTable(model)
          {
               private static final long serialVersionUID = 1L;

               public boolean isCellEditable(int row, int col-
umn)
               {
                    return false;
               };
          };
          table.getColumnModel().getColumn(0).setWidth(10);
          table.getColumnModel().getColumn(1).setWidth(20);
          table.getColumnModel().getColumn(2).setWidth(5);
          table.getColumnModel().getColumn(3).setWidth(10);

          //Where table containing a catalogue of all existing
sets is initialized
          //Reads from the LEGO_Collection.db file
          String query = "SELECT set_num, name, year, theme_id,
num_parts FROM LEGO_Database";

          // Establishes connection to the database
          SQLiteDataSource source = catalogue.getDs();

          try (Connection conn = source.getConnection(); State-
ment stmt = conn.createStatement();) {
               ResultSet rs = stmt.executeQuery(query);
               while (rs.next())
               {
                    model.insertRow(model.getRowCount(), new
String[] { rs.getString("set_num"), rs.getString("name"),
                              rs.getString("year"),
rs.getString("theme_id"), rs.getString("num_parts") });
               }
          }
          catch (SQLException e)
          {
               e.printStackTrace();
               System.exit(0);
```

```
          }
          table.setRowSelectionAllowed(true);
          JScrollPane scrollPane = new JScrollPane(table);
          scrollPane.setBounds(98, 129, 532, 230);
          contentPane.add(scrollPane);

          //Lets user search for specific sets using the con-
tents of the filterField
          //To add to the query String
          JButton searchBttn = new JButton("Search");
          searchBttn.setFont(new Font("Tahoma", Font.PLAIN,
14));
          searchBttn.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    String text = filterField.getText();
                    String query;
                    if (!filterField.getText().con-
tentEquals(""))
                    {
                         query = "SELECT set_num, name, year,
theme_id, num_parts FROM LEGO_Database WHERE name LIKE '%"
                                   + text + "%'";
                    }
                    else
                    {
                         query = "SELECT set_num, name, year,
theme_id, num_parts FROM LEGO_Database";
                    }

                    SQLiteDataSource source = catalogue.getDs();

                    try (Connection conn = source.getConnec-
tion(); Statement stmt = conn.createStatement();)
                    {
                         ResultSet rs = stmt.exe-
cuteQuery(query);
                         DefaultTableModel mod = (DefaultTable-
Model) table.getModel();
                         mod.setRowCount(0);
                         while (rs.next())
                         {
                              model.insertRow(model.getRow-
Count(),
```

```
                                        new String[]
{ rs.getString("set_num"), rs.getString("name"),
rs.getString("year"),

     rs.getString("theme_id"), rs.getString("num_parts") });
                         }
                    }
                    catch (SQLException j)
                    {
                         j.printStackTrace();
                         System.exit(0);
                    }
               }
          });
          searchBttn.setBounds(333, 80, 84, 31);
          contentPane.add(searchBttn);

          //Where the user enters the quantity of a particular
set they have in their collection
          JLabel qtyLbl = new JLabel("Quantity:");
          qtyLbl.setFont(new Font("Tahoma", Font.PLAIN, 14));
          qtyLbl.setBounds(98, 414, 65, 21);
          contentPane.add(qtyLbl);

          qtyField = new JTextField();
          qtyField.setFont(new Font("Tahoma", Font.PLAIN, 14));
          qtyField.setColumns(10);
          qtyField.setBounds(98, 445, 65, 39);
          contentPane.add(qtyField);

          //Where user inputs the purchase date of the set they
obtained
          JLabel dateLabel = new JLabel("Date Purchased:");
          dateLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
          dateLabel.setBounds(98, 382, 114, 22);
          contentPane.add(dateLabel);

          //Initializes a datePicker
          //Which lets the user choose the date they purchased a
set from an
          //Interactive calendar GUI
          UtilDateModel dModel = new UtilDateModel();
          Properties p = new Properties();
          p.put("text.today", "Today");
          p.put("text.month", "Month");
          p.put("text.year", "Year");
```

```
        JDatePanelImpl datePanel = new JDatePanelImpl(dModel,
p);
        JDatePickerImpl datePicker = new JDatePickerImpl(date-
Panel, new DateLabelFormatter());
        datePicker.getJFormattedTextField().setFont(new
Font("Tahoma", Font.PLAIN, 14));
        datePicker.setBounds(214, 376, 170, 40);
        contentPane.add(datePicker);

        //Where the user inputs the purchase price of the set
they obtained
        JLabel priceLabel = new JLabel("Purchase Price (num-
bers only):");
        priceLabel.setFont(new Font("Tahoma", Font.PLAIN,
14));
        priceLabel.setBounds(214, 414, 246, 22);
        contentPane.add(priceLabel);

        purPrice = new JTextField();
        purPrice.setFont(new Font("Tahoma", Font.PLAIN, 14));
        purPrice.setColumns(10);
        purPrice.setBounds(214, 445, 170, 39);
        contentPane.add(purPrice);

        //User can enter notes about the set they purchased
here if they wish
        JLabel notesFieldLabel = new JLabel("Notes:");
        notesFieldLabel.setFont(new Font("Tahoma", Font.PLAIN,
14));
        notesFieldLabel.setBounds(98, 494, 49, 31);
        contentPane.add(notesFieldLabel);

        notesField = new JTextField();
        notesField.setHorizontalAlignment(SwingCon-
stants.LEFT);
        notesField.setFont(new Font("Tahoma", Font.PLAIN,
14));
        notesField.setColumns(10);
        notesField.setBounds(98, 524, 532, 86);
        contentPane.add(notesField);

        // Adds information user inputted, as well as the se-
lected set info
        // To the Collection table under the user's userId
        JButton addButton = new JButton("Add");
        addButton.addMouseListener(new MouseAdapter()
        {
```

```
                @Override
                public void mouseClicked(MouseEvent e)
                {
                     int row = table.getSelectedRow();
                     //Makes sure that a row is selected, dis-
plays error otherwise
                     if (row >= 0)
                     {
                          setNum = table.getValueAt(row,
0).toString();
                          //Displays error if any of the fields
are null
                          if (datePicker.getJFormattedText-
Field().getText().contentEquals("") || qtyField.getText().con-
tentEquals("") || purPrice.getText().contentEquals(""))
                          {
                               errorLbl.setVisible(true);
                               errorLbl.setText("Please fill out
all fields.");
                          }
                          else
                          {
                               int qTest = 0;
                               double pTest;
                               // Makes sure quantity and price
are appropriate values (int and double)
                               try
                               {
                                    qTest = Integer.parseInt(qty-
Field.getText());
                                    quantity = qTest;

                                    pTest = Double.parseDou-
ble(purPrice.getText());
                                    NumberFormat formatter = Num-
berFormat.getCurrencyInstance(); // Casts the price to a mone-
tary

                                                          // value
                                    price = "" + formatter.for-
mat(pTest) + "";
                                    date = datePicker.getJFormat-
tedTextField().getText();
                                    notes = notesField.getText();
```

```
                                        Collection cur = new Collec-
tion(userId, quantity, setNum, table.getValueAt(row,
1).toString(),
                                                Integer.par-
seInt(table.getValueAt(row, 2).toString()), date, price, notes);
                                        cur.addEntry(); // Adds the
information by writing to the LEGO_Collection.db file
                                        ViewWindow view = new View-
Window(); // Takes user back to ViewWindow to see the update to
their

                    // collection
                                        setVisible(false);
                                        view.setVisible(true);
                                        view.run(userId, fName, last-
Name, username, password);
                                   }
                                   catch (Exception p)
                                   {
                                        errorLbl.setVisible(true);
                                        errorLbl.setText("ERROR: In-
correct value type for quantity and/or price.");
                                   }
                              }
                    }
                    else
                    {
                         errorLbl.setVisible(true);
                         errorLbl.setText("Set not selected.
Please select a set.");
                    }
               }
          });
          addButton.setFont(new Font("Tahoma", Font.PLAIN, 16));
          addButton.setBounds(264, 629, 170, 55);
          contentPane.add(addButton);

          // Allows user to sort entries based on column headers
          // Entries sorted in ascending order
          JLabel sortInstructions = new JLabel("Sort by: ");
          sortInstructions.setBounds(427, 78, 84, 34);
          sortInstructions.setFont(new Font("Tahoma",
Font.PLAIN, 14));
          sortInstructions.setHorizontalAlignment(SwingCon-
stants.CENTER);
          contentPane.add(sortInstructions);
```

```
          JComboBox<String> sortOptions = new JCom-
boBox<String>();
          sortOptions.setBounds(505, 81, 114, 29);
          sortOptions.setFont(new Font("Tahoma", Font.PLAIN,
14));
          sortOptions.setEnabled(true);
          sortOptions.addItem("Set #");
          sortOptions.addItem("Name");
          sortOptions.addItem("Year Released");
          sortOptions.addItem("# of Parts");
          sortOptions.addActionListener(new ActionListener()
          {
               public void actionPerformed(ActionEvent e)
               {
                    //Gets the String selected from the sortOp-
tions box
                    String sort = sortOptions.getSelectedI-
tem().toString();

                    //Based on the contents of sort
                    //Changes selectedSort to the appropriate
column header to use in the query String
                    switch (sort)
                    {
                    case "Set #":
                         selectedSort = "set_num";
                         break;
                    case "Name":
                         selectedSort = "name";
                         break;
                    case "Year":
                         selectedSort = "year";
                         break;
                    case "# of Parts":
                         selectedSort = "num_parts";
                         break;
                    }

                    // Deletes table's contents and reinitial-
izes them using the given specifications in
                    // the query String
                    String query = "SELECT set_num, name, year,
theme_id, num_parts FROM LEGO_Database ORDER BY "
                              + selectedSort + " ASC";
                    SQLiteDataSource source = catalogue.getDs();
```

```
                    try (Connection conn = source.getConnec-
tion(); Statement stmt = conn.createStatement();)
                    {

                         ResultSet rs = stmt.exe-
cuteQuery(query);

                         DefaultTableModel mod = (DefaultTable-
Model) table.getModel();
                         mod.setRowCount(0);
                         while (rs.next())
                         {
                              model.insertRow(model.getRow-
Count(),
                                        new String[]
{ rs.getString("set_num"), rs.getString("name"),
rs.getString("year"),

     rs.getString("theme_id"), rs.getString("num_parts") });
                         }
                    }
                    catch (SQLException r)
                    {
                         r.printStackTrace();
                         System.exit(0);
                    }
               }
          });
          contentPane.add(sortOptions);

          // Cancels the user's adding process,
          //Closes the Add Entry Window
          //Displays the View Window
          JButton cancelButton = new JButton("Cancel");
          cancelButton.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    ViewWindow view = new ViewWindow();
                    setVisible(false);
                    view.setVisible(true);
                    view.run(userId, fName, lastName, username,
password);
               }
          });
```

```
            cancelButton.setFont(new Font("Tahoma", Font.PLAIN,
16));
            cancelButton.setBounds(460, 629, 170, 54);
            contentPane.add(cancelButton);
      }

      /**
       * Create the frame.
       */
      public AddEntry()
      {


      }

}
```

Edit Entry Code

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;

import org.jdatepicker.impl.JDatePanelImpl;
import org.jdatepicker.impl.JDatePickerImpl;
import org.jdatepicker.impl.UtilDateModel;
import org.sqlite.SQLiteDataSource;

import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement; //?
import java.text.NumberFormat;
import java.util.Properties;

import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import java.awt.event.MouseAdapter;
import javax.swing.JButton;
import java.awt.Color;

public class EditEntry extends JFrame
{

     private JPanel contentPane;
     private JTextField qtyField;
     private JTextField purPrice;
     private JTextField notesField;
     public String fName;
     public String lastName;
     public int userId;
```

```
     public String username;
     public String password;
     public int quantity;
     public String setNum;
     public String name;
     public int year;
     public String date;
     public String price;
     public String notes;

     /**
      * Launch the application.
      */
     public static void main(String[] args)
     {
          EventQueue.invokeLater(new Runnable()
          {
               public void run()
               {
                    try
                    {
                         EditEntry frame = new EditEntry();
                         frame.setVisible(true);
                    }
                    catch (Exception e)
                    {
                         e.printStackTrace();
                    }
               }
          });
     }

     // Main method where everything is run; Otherwise certain
variables would be
     // null
     // causing the program to crash
     public void run(int id, String first, String last, String
use, String pas, int qty, String set, String setName,
               int release, String purDate, String purchase,
String descr)
     {
          userId = id;
          fName = first;
          lastName = last;
          username = use;
          password = pas;
          quantity = qty;
```

```
            setNum = set;
            name = setName;
            year = release;
            date = purDate;
            price = purchase;
            notes = descr;

            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(100, 100, 500, 500);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(null);

            // Instruction label for editing entries
            JLabel lblEditInformationBelow = new JLabel("Edit in-
formation below.");
            lblEditInformationBelow.setHorizontalAlign-
ment(SwingConstants.CENTER);
            lblEditInformationBelow.setFont(new Font("Tahoma",
Font.PLAIN, 16));
            lblEditInformationBelow.setBounds(81, 22, 336, 22);
            contentPane.add(lblEditInformationBelow);

            //Displays the correct errors when appropriate
            JLabel errorLbl = new JLabel("");
            errorLbl.setForeground(Color.RED);
            errorLbl.setHorizontalAlignment(SwingConstants.CEN-
TER);
            errorLbl.setFont(new Font("Tahoma", Font.BOLD, 14));
            errorLbl.setBounds(47, 41, 401, 47);
            contentPane.add(errorLbl);
            errorLbl.setVisible(false);

            //Displays the set information being edited
            JLabel editingLbl = new JLabel("Editing: " + name); //
Okay idk how I'm doing this...
            editingLbl.setHorizontalAlignment(SwingCon-
stants.LEFT);
            editingLbl.setFont(new Font("Tahoma", Font.PLAIN,
14));
            editingLbl.setBounds(47, 85, 357, 34);
            contentPane.add(editingLbl);

            //Displays the quantity of a set being edited
            JLabel qtyLbl = new JLabel("Quantity:");
            qtyLbl.setFont(new Font("Tahoma", Font.PLAIN, 14));
```

```
            qtyLbl.setBounds(47, 168, 65, 21);
            contentPane.add(qtyLbl);

            qtyField = new JTextField();
            qtyField.setFont(new Font("Tahoma", Font.PLAIN, 14));
            qtyField.setColumns(10);
            qtyField.setBounds(47, 195, 65, 39);
            contentPane.add(qtyField);
            qtyField.setText("" + quantity + "");

            //Initializes a datePicker
            //Which lets the user choose the date they purchased a
set from an
            //Interactive calendar GUI
            JLabel dateLabel = new JLabel("Date Purchased:");
            dateLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
            dateLabel.setBounds(47, 119, 114, 39);
            contentPane.add(dateLabel);

            UtilDateModel dModel = new UtilDateModel();
            Properties p = new Properties();
            p.put("text.today", "Today");
            p.put("text.month", "Month");
            p.put("text.year", "Year");
            JDatePanelImpl datePanel = new JDatePanelImpl(dModel,
p);
            JDatePickerImpl datePicker = new JDatePickerImpl(date-
Panel, new DateLabelFormatter());
            datePicker.getJFormattedTextField().setFont(new
Font("Tahoma", Font.PLAIN, 14));
            datePicker.setBounds(152, 129, 170, 40);
            contentPane.add(datePicker);

            // The purchase price of a set
            // Displays the purchase price
            JLabel priceLbl = new JLabel("Purchase Price:");
            priceLbl.setFont(new Font("Tahoma", Font.PLAIN, 14));
            priceLbl.setBounds(152, 167, 95, 22);
            contentPane.add(priceLbl);

            purPrice = new JTextField();
            purPrice.setFont(new Font("Tahoma", Font.PLAIN, 14));
            purPrice.setColumns(10);
            purPrice.setBounds(152, 195, 170, 39);
            contentPane.add(purPrice);
            purPrice.setText(price.substring((price.indexOf("$") +
1), price.length()));
```

```
          //User can edit their notes about the set they pur-
chased here if they wish
          JLabel notesFieldLabel = new JLabel("Notes:");
          notesFieldLabel.setFont(new Font("Tahoma", Font.PLAIN,
14));
          notesFieldLabel.setBounds(47, 244, 49, 31);
          contentPane.add(notesFieldLabel);

          notesField = new JTextField();
          notesField.setHorizontalAlignment(SwingCon-
stants.LEFT);
          notesField.setFont(new Font("Tahoma", Font.PLAIN,
14));
          notesField.setColumns(10);
          notesField.setBounds(47, 285, 391, 86);
          contentPane.add(notesField);
          notesField.setText(notes);

          //Closes Edit Entry Window
          //Opens View Window
          JButton cancelButton = new JButton("Cancel");
          cancelButton.setFont(new Font("Tahoma", Font.PLAIN,
14));
          cancelButton.setBounds(298, 381, 140, 40);
          cancelButton.addMouseListener(new MouseAdapter()
          {
               @Override
               public void mouseClicked(MouseEvent e)
               {
                    ViewWindow view = new ViewWindow();
                    setVisible(false);
                    view.setVisible(true);
                    view.run(userId, fName, lastName, username,
password);
               }
          });
          contentPane.add(cancelButton);

          // Saves the edits to the Collection table in
LEGO_Databse
          JButton saveButton = new JButton("Save");
          saveButton.setFont(new Font("Tahoma", Font.PLAIN,
14));
          saveButton.setBounds(148, 381, 140, 40);
          saveButton.addMouseListener(new MouseAdapter()
          {
```

```
                @Override
                public void mouseClicked(MouseEvent e)
                {
                     int qTest = 0;
                     double pTest;
                     quantity = 0;
                     price = null;
                     //Checks to make sure all fields are filled
out
                     if (datePicker.getJFormattedText-
Field().getText().contentEquals("") || qtyField.getText().con-
tentEquals("") ||
                               purPrice.getText().con-
tentEquals(""))
                     {
                          errorLbl.setVisible(true);
                          errorLbl.setText("Please fill out all
fields.");
                     }
                     else
                     {
                          try // Makes sure that quantity and
price are appropriate values (int and double)
                          {
                               qTest = Integer.parseInt(qty-
Field.getText());
                               quantity = qTest;

                               pTest = Double.parseDouble(pur-
Price.getText());
                               NumberFormat formatter = Number-
Format.getCurrencyInstance(); // Makes it so that the entered

                                                    // String is a
monetary value
                               price = "" + formatter.for-
mat(pTest) + "";
                               date = datePicker.getJFormat-
tedTextField().getText();
                               notes = notesField.getText();

                               Collection cur = new Collec-
tion(userId, quantity, setNum, name, year, date, price, notes);
                               cur.editEntry(); // Updates the
information by writing to the LEGO_Collection.db file
```

```
                              ViewWindow view = new ViewWin-
dow(); // Takes user back to ViewWindow to see the update to
their

               // collection
                              setVisible(false);
                              view.setVisible(true);
                              view.run(userId, fName, lastName,
username, password);
                         }
                         catch (Exception p)
                         {
                              errorLbl.setVisible(true);
                              errorLbl.setText("ERROR: Incorrect
value type for quantity and/or price.");
                         }
                    }
               }

          });
          contentPane.add(saveButton);
     }

     /*
      * /** Create the frame.
      */
     // This class is empty because otherwise necessary varia-
bles (userId, fName,
     // lastName, etc. would not be carried over,
     // causing the program to crash)
     public EditEntry()
     {

     }
}
```

Users

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import org.sqlite.SQLiteDataSource;

//Class for writing to the Users table in the LEGO_Collection
Database
//Utilizes SQLite
public class Users
{
     private int myUserId;
     private String myFName;
     private String myLName;
     private String myUsername;
     private String myPassword;
     private Database myUsers;

     public Users(int userId, String fName, String lastName,
String user, String pas)
     {
          myUserId = userId;
          myFName = fName;
          myLName = lastName;
          myUsername = user;
          myPassword = pas;
          myUsers = new Database();
     }

     //Returns a Users object if it exists
     //Accomplishes this by reading from the LEGO_Collection.db
file
     public Users getUser(String username, String password)
     {
          String query = "SELECT user_id, first_name, last_name,
username, password FROM Users WHERE username = '"
                    + myUsername + "' AND password = '" +
myPassword + "'";

          //Establishes database connection
          SQLiteDataSource source = myUsers.getDs();
          Users usr = null;
          try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();)
```

```
          {
               ResultSet rs = stmt.executeQuery(query);
               //if rs.isClosed(), it means that the requested
information from the LEGO_Collection.db file does not exist
               if (rs.isClosed() == false)
               {
                    usr = new Users(rs.getInt("user_id"),
rs.getString("first_name"), rs.getString("last_name"),
                              rs.getString("username"),
rs.getString("password"));
               }
          }
          catch (SQLException e)
          {
               e.printStackTrace();
               System.exit(0);
          }
          return usr;
     }

     //Checks if the entered username is taken when creating a
new account
     //By reading from the LEGO_Collection.db file
     public boolean isUsernameTaken(String username)
     {
          String query = "SELECT user_id FROM Users WHERE
username = '" + myUsername + "'";

          //Establishes database connection
          SQLiteDataSource source = myUsers.getDs();
          try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();)
          {
               ResultSet rs = stmt.executeQuery(query);
               //if rs.isClosed(), it means that the requested
information from the LEGO_Collection.db file does not exist
               if (rs.isClosed() == false)
               {
                    return true;
               }
          }
          catch (SQLException e)
          {
               e.printStackTrace();
               System.exit(0);
          }
          return false;
```

```
     }

     //Creates a new user and adds it to the database
     //By writing to the LEGO_Collection.db file
     public void addUser()
     {
          String query1 = "INSERT INTO Users(username, password,
first_name, last_name) VALUES ('" + myUsername + "', '"
                    + myPassword + "', '" + myFName + "', '" +
myLName + "')";
          SQLiteDataSource source = myUsers.getDs();

          try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();)
          {
               stmt.execute(query1); // inserts new user into
Users table; unique user_id generated when this occurs,
                                                  // which is why
myUserId is not used here
          }
          catch (SQLException e)
          {
               e.printStackTrace();
               System.exit(0);
          }
     }

     //Returns the user's userId
     public int getUserId()
     {
          return myUserId;
     }

     //Sets user's userId
     public void setUserId(int id)
     {
          myUserId = id;
     }

     // Returns user's first name
     public String getFirstName()
     {
          return myFName;
     }

     //Sets user's first name
     public void setFirstName(String first)
```

```
    {
        myFName = first;
    }

    // Returns user's last name
    public String getLastName()
    {
        return myLName;
    }

    //Sets user's last name
    public void setLastName(String last)
    {
        myLName = last;
    }

    // Returns username;
    public String getUsername()
    {
        return myUsername;
    }

    //Sets username
    public void setUsername(String use)
    {
        myUsername = use;
    }

    // Returns password;
    public String getPassword()
    {
        return myPassword;
    }

    //Sets password
    public void setPassword(String pas)
    {
        myPassword = pas;
    }

}
```

Collection

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.table.DefaultTableModel;

import org.sqlite.SQLiteDataSource;

//Class for writing to the Collection table in the
LEGO_Collection Database
//Utilizes SQLite

public class Collection
{
     private int myUserId;
     private int myQuantity;
     private String mySetNum;
     private String myName;
     private int myYear;
     private String myPurDate;
     private String myPrice;
     private String myNotes;
     private Database myCollection;

     public Collection(int id, int qty, String num, String
setName, int release, String date, String purPrice,
                String note)
     {
          myUserId = id;
          myQuantity = qty;
          mySetNum = num;
          myName = setName;
          myYear = release;
          myPurDate = date;
          myPrice = purPrice;
          myNotes = note;
          myCollection = new Database();
     }

     //Adds an entry into the Collection table by writing to the
LEGO_Collection.db file
     //The user_id denotes which user's set that is
     public void addEntry()
```

```
     {
          String query = "INSERT INTO Collection(user_id,
quantity, set_num, name, year, pur_date, price, notes) VALUES "
                    + "(" + myUserId + ", " + myQuantity + ", '"
+ mySetNum + "', '" + myName + "', " + myYear + ", '"
                    + myPurDate + "', '" + myPrice + "', '" +
myNotes + "')";

          SQLiteDataSource source = myCollection.getDs();

          try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();) {
               stmt.execute(query);
               System.out.println(query);

          }
          catch (SQLException e)
          {
               e.printStackTrace();
               System.exit(0);
          }
     }

     //Adds edited information to a particular row in the
Collection table
     //By writing to the LEGO_Collection.db file where specified
     //Specific row denoted by the userId and the set number
given
     public void editEntry()
     {
          String query = "UPDATE Collection SET quantity = " +
myQuantity + ", " + "pur_date = '" + myPurDate + "', "
                    + "price = '" + myPrice + "', " + "notes =
'" + myNotes + "' " + "WHERE user_id = " + myUserId
                    + " AND set_num = '" + mySetNum + "'";

          //Establishes database connection
          SQLiteDataSource source = myCollection.getDs();

          try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();)
          {
               stmt.execute(query);

          }
          catch (SQLException p)
          {
```

```
                p.printStackTrace();
                System.exit(0);
            }
        }

        //Deletes information to a particular row in the Collection
table
        //By deleting from the LEGO_Collection.db file where
specified
        //Specific row denoted by the userId and the set number
given
        public void deleteEntry()
        {
            String query = "DELETE FROM Collection WHERE user_id =
" + myUserId + " AND set_num = '" + mySetNum + "'";

            //Establishes database connection
            SQLiteDataSource source = myCollection.getDs();

            try (Connection conn = source.getConnection();
Statement stmt = conn.createStatement();)
            {
                stmt.execute(query);
            }
            catch (SQLException r)
            {
                r.printStackTrace();
                System.exit(0);
            }
        }
}
```

Database

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import org.sqlite.SQLiteDataSource;

public class Database
{
     private SQLiteDataSource ds;

     //Initializes database by specifying the URL of the file
     public Database()
     {
          try
          {
               ds = new SQLiteDataSource();
               ds.setUrl("jdbc:sqlite:LEGO_Collection.db");
          }
          catch (Exception e)
          {
               e.printStackTrace();
               System.exit(0);
          }
     }

     //Returns ds
     public SQLiteDataSource getDs()
     {
          return ds;
     }
}
```

Date Label Formatter Code

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import javax.swing.JFormattedTextField.AbstractFormatter;

//Class initialies the DateLabel Formatter information used in
DatePicker (the Calendar date-picking GUI)
public class DateLabelFormatter extends AbstractFormatter
{

    private String datePattern = "yyyy-MM-dd";
    private SimpleDateFormat dateFormatter = new
SimpleDateFormat(datePattern);

    //Parses the String parameter to an object
    @Override
    public Object stringToValue(String text) throws
ParseException
    {
        return dateFormatter.parseObject(text);
    }

    //Does... Something
    @Override
    public String valueToString(Object value) throws
ParseException
    {
        if (value != null)
        {
            Calendar cal = (Calendar) value;
            return dateFormatter.format(cal.getTime());
        }

        return "";
    }

}
```