SDF Library 3
December 6th, 2015
Kyra Koch

Account.java

```java
/**
 * Individual account
 *
 * @author Kyra Koch
 * @version 2.0
 */

package cu.cs.cpsc2150.project2;

import java.io.Serializable;


public class Account implements Comparable<Account>, Serializable {
    private int myID;
    private String myName, myUsername, myPassword, myEmail, myPhone;
    private boolean staff;

    /**
     * construct for the account
     * @param id              the id for the new account
     * @param name            the new account user's name
     * @param username the new account's username
     * @param password the new account's password
     * @param email           the new account's email
     * @param phone           the new account's phone number
     * @param staff           whether or not it is a staff account, if staff, variable will be true
     */
    public Account(int id, String name, String username, String password, String email, String
phone, boolean staff) {
        this.myID = id;
        this.myName = name;
        this.myUsername = username;
        this.myPassword = password;
        this.staff = staff;
        this.myEmail = email;
        this.myPhone = phone;
    }

    /**
     * returns ID
     * @return               account's ID
     */
    public int getID() {
        return myID;
    }

    /**
     * returns username
     * @return               account's username
     */
    public String getUsername() {
        return myUsername;
    }

    /**
     * returns true if staff account
     * @return               staff
     */
    public boolean getStaff() {
       return staff;
    }

    /**
     * returns user's name
     * @return               user's name
     */
    public String getMyName() {
       return myName;
    }

    /**
     * checks to make sure username and password match
     * @param username the username to check with this username
     * @param password the password to check with the username
```

```
     * @return               if password matches, return true
     */
    public boolean authenticate(String username, String password) {
        return myUsername.equals(username) && myPassword.equals(password);
    }

        /**
         * compares current account ID with given account ID
         * @param o            given account
         * @return             returns standard compareTo return values
         */
    @Override
    public int compareTo(Account o) {
        return ((Integer)myID).compareTo(o.myID);
    }

    /**
     * returns the user's name
     * @return the user's name
     */
    @Override
    public String toString() {
        return myName;
    }

    /**
     * returns the user's password
     * @return the user's password
     */
       public String getPassword() {
              return myPassword;
       }

       /**
        * returns the account type as a String
        * @return     the account type as a String
        */
       public String getType() {
              if(staff)
                     return "Staff";
              else
                     return "Member";
       }
}
```

AccountDatabase.java

```java
/**
 * AccountDatabase class holds all of the accounts and performs action on them
 *
 * @author Kyra Koch
 * @version 2.0
 */

package cu.cs.cpsc2150.project2;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;

public class AccountDatabase implements Serializable {
    private ArrayList<Account> myAccounts;

    /**
     * Initial AccountDatabase hold one account: Administrator
     */
    public AccountDatabase() {
        myAccounts = new ArrayList<>();
        myAccounts.add(new Account(0, "Administrator", "admin", "pass", "", "", true));
    }

    /**
     * adds an account to the database
     * @param account  the account to add
     */
    public void addAccount(Account account) {
        myAccounts.add(account);
        Collections.sort(myAccounts);
    }

    /**
     * removes an account from the database
     * @param id              the ID of the account to remove
     */
    public void removeAccount(int id) {
        for (int i = 0; i < myAccounts.size(); ++i) {
            if (myAccounts.get(i).getID() == id) {
                myAccounts.remove(i);
                return;
            }
        }
    }

    /**
     * loops through all of the accounts to see there is an account with a given username password
combo
     * @param username the username to find
     * @param password the password o test
     * @return                returns the account if found
     */
    public Account login(String username, String password) {
        Account login = null;
        for (Account account : myAccounts) {
            if (account.authenticate(username, password)) {
                login = account;
                break;
            }
        }
        return login;
    }

    /**
     * returns a String of all of the usernames in the database
     * @return String of all of the usernames in the database
     */
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < myAccounts.size(); ++i) {
            sb.append(i);
            sb.append(") ");
```

```
            sb.append(myAccounts.get(i).getUsername());
            sb.append("\n");
        }
        return sb.toString();
    }

    /**
     * returns the size of the database
     * @return how many accounts are in the database
     */
       public int getSize() {
              return myAccounts.size();
       }

       /**
        * returns the account at a specific index
        * @param ndx  the index at which to return
        * @return            the account at the index
        */
       public Account getAccount(int ndx) {
              return myAccounts.get(ndx);
       }
}
```

Book.java

```java
/**
 * Book class contains all of the necessary info about a book, as well as the account that checks it
out
 *
 * @author Kyra Koch
 * @version 2.0
 */

package cu.cs.cpsc2150.project2;

import java.io.Serializable;
import java.util.ArrayList;

public class Book implements Comparable<Book>, Serializable {
    private int myID;
    private String myTitle, myAuthor, myGenre;
    private ArrayList<String> myTags;
    private Account acc;

    /**
     * creates an instance of a Book object
     * @param id              the ID of the new book, unique
     * @param title           the title of the book
     * @param author   the author of the book
     * @param genre           the genre of the book
     * @param tags            the tags of the new book
     */
    public Book(int id, String title, String author, String genre, ArrayList<String> tags) {
        this.myID = id;
        this.myTitle = title;
        this.myAuthor = author;
        this.myGenre = genre;
        this.myTags = tags;
        this.acc = null;
    }

    /**
     * returns the id of the book
     * @return the id of the book
     */
    public int getID() {
        return myID;
    }

    /**
     * sets the account that has checked out this book
     * @param acc      the account that has this book checked out
     */
    public void setAccount(Account acc) {
       this.acc = acc;
    }

    /**
     * returns the account that has this book checked out
     * @return the account that has this book checked out
     */
    public Account getAccount() {
       return acc;
    }

    /**
     * compare the two ids and return what the original compareTo does
     */
    @Override
    public int compareTo(Book o) {
        return ((Integer)myID).compareTo(o.myID);
    }

    /**
     * compares a given string to every field within the book and returns if it matches
     * used for suzzy search
     * @param in       the string to compare
     * @return         true if a field within the book matches the given string roughly
     */
```

```
    public boolean myCompare(String in) {
       if(in.toLowerCase().equals(myTitle.toLowerCase())||
in.toLowerCase().equals(myGenre.toLowerCase())
                       || in.toLowerCase().equals(myAuthor.toLowerCase())) {
               return true;
       }
       else
       {
               for(int i = 0; i < myTags.size(); i++) {
                       if(myTags.get(i).contains(in)) {
                               return true;
                       }
               }
       }
               return false;
    }

    /**
     * compares a given string to every field within the book and returns if it matches
     * used for literal search
     * @param in      the input string to compare
     * @return        true if a field within the book matches the given string exactly
     */
    public boolean myLitCompare(String in) {
       if(in.toLowerCase().equals(myTitle.toLowerCase())||
in.toLowerCase().equals(myGenre.toLowerCase())
                       || in.toLowerCase().equals(myAuthor.toLowerCase())) {
               return true;
       }
       else
       {
               for(int i = 0; i < myTags.size(); i++) {
                       if(myTags.get(i).equals(in)) {
                               return true;
                       }
               }
       }
               return false;
    }

    /**
     * prints the tags of this book
     * @return a single string of all of the tags separated by commas
     */
    public String printTags()
    {
               String retStr = "";
               for(int i = 0; i < myTags.size(); i++)
                       retStr = retStr + " " + myTags.get(i);
               return retStr;
    }

    /**
     * prints out the relevant data on the book
     *
     * @return <Title>, by <Author> \nGenre: <Genre> \nTags: <Tags>
     */
    @Override
    public String toString() {
        return myTitle + ", by " + myAuthor + "\nGenre: " + myGenre + "\nTags: " + printTags();
    }

    /**
     * returns the id of the book
     * @return the id of the book
     */
       public int getMyID() {
               return myID;
       }

       /**
        * returns the title of the book
        * @return     the title of the book
        */
       public String getMyTitle() {
               return myTitle;
       }
```

```java
	/**
	 * returns the author of the book
	 * @return	the author of the book
	 */
	public String getMyAuthor() {
		return myAuthor;
	}

	/**
	 * returns the genre of the book
	 * @return	the genre of the book
	 */
	public String getMyGenre() {
		return myGenre;
	}

	/**
	 * returns the tags as an arraylist
	 * @return	the tags as an arraylist
	 */
	public ArrayList<String> getMyTags() {
		return myTags;
	}

	/**
	 * checks whether or not a book is checked out
	 * @return	true if the book is checked out, false if otherwise
	 */
	public boolean checkedout() {
		if(acc == null)
			return false;
		else
			return true;
	}
}
```

Catalog.java

```java
/**
 * Holds all of the Books in the Catalog
 *
 * @author Kyra Koch
 * @version 2.0
 */

package cu.cs.cpsc2150.project2;

import java.io.Serializable;
import java.util.*;

public class Catalog implements Serializable {
    private ArrayList<Book> myBooks;

    /**
     * Creates a new Catalog
     */
       public Catalog() {
        myBooks = new ArrayList<Book>();
    }

       /**
        * Creates a new Catalog with a given list of books
        * @param book the list of Books to be added
        */
       public Catalog(ArrayList<Book> book) {
              myBooks = book;
       }

       /**
        * adds a Book to the Catalog
        * @param book the Book to add to the Catalog
        */
    public void addBook(Book book) {
        myBooks.add(book);
        Collections.sort(myBooks);
    }

    /**
     * removes a Book from the Catalog
     * @param id       the ID of the Book to remove
     */
    public void removeBook(int id) {
        for (int i = 0; i < myBooks.size(); ++i) {
            if (myBooks.get(i).getID() == id) {
                myBooks.remove(i);
                return;
            }
        }
    }

    /**
     * returns all the Books in the Catalog
     * @return
     */
    public ArrayList<Book> getMyBooks() {
              return myBooks;
       }

    /**
     * returns the Book of a given ID
     * @param ID       the Id of the book to find
     * @return         the Book with a given ID
     */
    public Book findBook(int ID) {
       for(int i = 0; i < myBooks.size(); i++) {
              if(myBooks.get(i).getID() == ID) {
                     return myBooks.get(i);
              }
       }
       return null;
    }
```

```
    /**
     * returns a list of all Books in the Catalog
     */
       @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < myBooks.size(); ++i) {
            sb.append(i);
            sb.append(") ");
            sb.append(myBooks.get(i));
            sb.append("\n");
        }
        return sb.toString();
    }

        /**
         * returns the size of the catalog
         * @return     the number of entries in the Catalog
         */
        public int getSize() {
               return myBooks.size();
        }

        /**
         * returns the Book of a given index
         * @param ndx  the index of the Book to get
         * @return             the Book at a given index
         */
        public Book getBook(int ndx) {
               return myBooks.get(ndx);
        }

        /**
         * Finds all of the Books a given Account has checked out
         * @param acc  the Account to find the Books for
         * @return             an ArrayList of all the Books the given Account has checked out
         */
        public ArrayList<Book> userBooks(Account acc) {
               ArrayList<Book> arr = new ArrayList<Book> ();
               for (int i = 0; i < myBooks.size(); ++i) {
            if(myBooks.get(i).getAccount() != null && myBooks.get(i).getAccount().getID() ==
acc.getID())
               arr.add(myBooks.get(i));
        }
               return arr;
        }
}
```

Cart.java

```
/**
 * The Cart class holds instances of CartItems
 *
 * @author Kyra Koch
 * @version 1.0
 */
package cu.cs.cpsc2150.project2;

import java.util.ArrayList;

public class Cart {
        ArrayList<CartItem> cart;

        /**
         * Constructor of the Cart
         */
        public Cart() {
               cart = new ArrayList<CartItem>();
        }

        /**
         * creates a new CartItem with the command CHECKOUT
         * @param b            the Book to be checked out
         * @param acc  the Account with which to check it out to
         */
        public void checkoutBook(Book b, Account acc) {
               CartItem it = new CartItem(b, acc, CartCommand.CHECKOUT);
```

```
                cart.add(it);
        }

        /**
         * creates a new CartItem with the command RETURN
         * @param b            the Book to be returned
         * @param acc  the Account with whom it was checked out
         */
        public void returnBook(Book b, Account acc) {
                CartItem it = new CartItem(b, acc, CartCommand.RETURN);
                cart.add(it);
        }

        /**
         * Actually checks out and/or returns the Books
         */
        public void completeCheckout() {
                for(int i = 0; i < cart.size(); i++) {
                        cart.get(i).complete();
                }
                cart.clear();
        }

        /**
         * returns the number of items in the cart
         * @return     the number of items in the cart
         */
        public int getSize() {
                return cart.size();
        }

        /**
         * returns the CartItem at a specified index
         * @param ndx  the index of the item you want
         * @return             the index
         */
        public CartItem get(int ndx) {
                return cart.get(ndx);
        }
}
```

```java
/**
 * The CartItem hold a Book, an Account, and a CartCommand
 *
 * @author Kyra Koch
 * @version 1.0
 */
package cu.cs.cpsc2150.project2;

public class CartItem {
	Book book;
	Account account;
	CartCommand command;

	/**
	 * Constructor for the CartItem
	 * @param b		the Book to perform that action on
	 * @param acc	the Account associated/to associated with Book
	 * @param comm the CartCommand
	 */
	public CartItem(Book b, Account acc, CartCommand comm) {
		this.book = b;
		this.account = acc;
		this.command = comm;
	}

	/**
	 * executes the given CartCommand
	 */
	public void complete() {
		if(command == CartCommand.RETURN) {
			if(book.getAccount() != account) {
				return;
			}
			else {
				book.setAccount(null);
				return;
			}
		}
		else if(command == CartCommand.CHECKOUT) {
			if(book.getAccount() != null) {
				return;
			}
			else {
				book.setAccount(account);
				return;
			}
		}
		else {
			System.out.println("Error! Something went wrong!");
			return;
		}
	}

	/**
	 * returns the book for CartItem
	 * @return	Book
	 */
	public Book getBook() {
		return book;
	}

	/**
	 * returns the CartCommand for CartItem
	 * @return	CartCommand
	 */
	public CartCommand getCommand() {
		return command;
	}
}
```

CartCommand.java

```java
/**
 * The CartCommands that can be performed on the Cart: CHECKOUT, RETURN, NONE
 *
 * @author Kyra Koch
 * @version 1.0
 */
package cu.cs.cpsc2150.project2;

public enum CartCommand {
        CHECKOUT,
        RETURN,
        NONE
}
```

Search.java

```java
/**
 * The interface by which FuzzySearch and LiteralSearch adhere
 *
 * @author Kyra Koch
 * @version 1.0
 */

package cu.cs.cpsc2150.project2;

import java.util.ArrayList;

public interface Search {
	public ArrayList<Book> search(String in, ArrayList<Book> cata);
}
```

FuzzySearch.java

```
/**
 * Performs a "fuzzy" search on a given catalog. Checks for plural and ignores case sensitivity
 * Employs the Singleton pattern
 *
 * @author Kyra Koch
 * @version 1.0
 */
package cu.cs.cpsc2150.project2;

import java.util.ArrayList;

public class FuzzySearch implements Search{
	private static final FuzzySearch fuzzy = new FuzzySearch();

	private FuzzySearch() {};

	/**
	 * uses Singleton Pattern
	 * @return     returns the instance of the fuzzy search
	 */
	public static FuzzySearch getInstance()
	{
		return fuzzy;
	}

	/**
	 * searches through a given catalog. Checks for plural and ignores case sensitivity
	 *
	 * @param in   the String with which to search by
	 * @param cata the Catalog to search
	 * @return             all of the Books that fit a given criteria
	 */
	public ArrayList<Book> search(String in, ArrayList<Book> cata) {
		ArrayList<Book> retList = new ArrayList<Book>();
	for(int i = 0; i < cata.size(); i++) {
		if(cata.get(i).myCompare(in))
		{
			retList.add(cata.get(i));
		}
		else
		{
			if(in.endsWith("s"))
			{
				if(in.endsWith("ies"))
				{
					if(cata.get(i).myCompare(in.substring(0, in.length()-
3)+"y"))
						retList.add(cata.get(i));
				}
				else
				{
					if(cata.get(i).myCompare(in.substring(0, in.length()-
1)))
						retList.add(cata.get(i));
				}
			}
		}
	}
	return retList;
	}
}
```

LiteralSearch.java

```java
/**
 * Performs a literal search on a given catalog. Ignores case sensitivity
 * Employs the Singleton pattern
 *
 * @author Kyra Koch
 * @version 1.0
 */
package cu.cs.cpsc2150.project2;

import java.util.ArrayList;

public class LiteralSearch implements Search {
	private static final LiteralSearch literal = new LiteralSearch();

	private LiteralSearch() {};

	/**
	 * uses Singleton Pattern
	 * @return     returns the instance of the literal search
	 */
	public static LiteralSearch getInstance()
	{
		return literal;
	}

	/**
	 * searches through a given catalog. Ignores case sensitivity
	 *
	 * @param in   the String with which to search by
	 * @param cata the Catalog to search
	 * @return             all of the Books that fit a given criteria
	 */
	public ArrayList<Book> search(String in, ArrayList<Book> cata) {
		ArrayList<Book> retList = new ArrayList<Book>();
	for(int i = 0; i < cata.size(); i++) {
		if(cata.get(i).myLitCompare(in))
		{
			retList.add(cata.get(i));
		}
	}
	return retList;
	}

}
```

Main.java

```
package cu.cs.cpsc2150.project3;

import java.io.FileNotFoundException;
import java.io.IOException;

public class Main {
	public static void main(String[] args) throws FileNotFoundException, ClassNotFoundException,
IOException {
		GUI.runGUI();
	}
}
```

GUI.java

```
/* Author: Blair Durkee
 * Editor: Kyra Koch*/

package cu.cs.cpsc2150.project3;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

import javax.swing.AbstractAction;
import javax.swing.ActionMap;
import javax.swing.ButtonGroup;
import javax.swing.InputMap;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.KeyStroke;
import javax.swing.ListSelectionModel;
import javax.swing.RowFilter;
import javax.swing.border.EmptyBorder;

import cu.cs.cpsc2150.project2.*;

public class GUI {
        private static AccountDatabase myAccountDatabase;
    private static Catalog myCatalog;
    private static Account myLoggedInAccount;
    private static Cart cart;
    private static JFrame login;
    private static JFrame appFrame;
    private static boolean loggedout;
    private static JTable acctbl;
    private static JTable catatbl;
    private static JTable usertbl;
    private static JTable actiontbl;
    private static AccountTable acctable;
    private static CatalogTable catatable;
    private static UserTable usertable;
    private static ActionTable actiontable;
    private static GridBagConstraints left;
    private static GridBagConstraints right;
    private static FuzzySearch fuzzy;
    private static LiteralSearch literal;

    public static void runGUI() throws FileNotFoundException, ClassNotFoundException, IOException {
```

```
    init();
    while(true) {
                    login();
                    login.setVisible(true);
                    while(myLoggedInAccount == null) {
                            System.out.print("");
                    }
                    loggedout = false;
                    login.setVisible(false);
                    login.dispose();

            appFrame();
            appFrame.setVisible(true);
            while(!loggedout) {
                    System.out.print("");
            }
            appFrame.setVisible(false);
            appFrame.dispose();
            try {
                            save();
                    } catch (Exception e1) {
                            //
                    }
    }
}

private static void init() throws FileNotFoundException, ClassNotFoundException, IOException {
    myAccountDatabase = new AccountDatabase();
    myCatalog = new Catalog();
    cart = new Cart();
    fuzzy = fuzzy.getInstance();
    literal = literal.getInstance();
    myLoggedInAccount = null;
    loggedout = true;
    left = new GridBagConstraints();
    left.anchor = GridBagConstraints.LINE_START;
    right = new GridBagConstraints();
    right.anchor = GridBagConstraints.LINE_END;
            right.weightx = 2.0;
            right.fill = GridBagConstraints.HORIZONTAL;
    right.gridwidth = GridBagConstraints.REMAINDER;

    File datFile = new File("input.dat");
    if(datFile.exists()) {
            load();
    }
    else {
            save();
    }
}

private static void login() {
    final JTextField nameF;
    final JPasswordField Password;
    final JLabel inval;

    login = new JFrame("Login");
    login.setSize(300, 112);
            login.setLocationRelativeTo(null);
            login.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            JPanel topPanel = new JPanel(new GridBagLayout());

            topPanel.add(new JLabel("Enter your username: "),left);
            nameF = new JTextField(20);
            topPanel.add(nameF, right);
            topPanel.add(new JLabel("Enter your password: "),left);
            Password = new JPasswordField(20);
            topPanel.add(Password, right);
            inval = new JLabel("Please enter valid username/password!");
    inval.setVisible(false);
    inval.setForeground(Color.red);
    topPanel.add(inval, right);

    int condition = JComponent.WHEN_FOCUSED;
    InputMap iMap = Password.getInputMap(condition);
    ActionMap aMap = Password.getActionMap();
```

```
        String enter = "enter";
        iMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0), enter);
        aMap.put(enter, new AbstractAction() {

           @Override
           public void actionPerformed(ActionEvent arg0) {
                  myLoggedInAccount = myAccountDatabase.login(nameF.getText(),
String.valueOf(Password.getPassword()));
                                if(myLoggedInAccount == null) {
                                        inval.setVisible(true);
                                }
           }
        });

                final JPanel windowPane = new JPanel(new BorderLayout());

                // create bottom panel
                final JPanel bottomPanel = new JPanel(new BorderLayout());
                JButton button1 = new JButton("Login");
                button1.setSize(30, 1);
                button1.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                try {
                                        myLoggedInAccount = myAccountDatabase.login(nameF.getText(),
String.valueOf(Password.getPassword()));
                                        if(myLoggedInAccount == null) {
                                                inval.setVisible(true);
                                        }
                                } catch (Exception e1) {
                                        //
                                }
                        }
                });
                JButton button2 = new JButton("Quit");
                button2.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent d) {
                                System.exit(0);
                        }
                });
                bottomPanel.add(BorderLayout.CENTER, button1);
                bottomPanel.add(BorderLayout.EAST, button2);


                windowPane.add(BorderLayout.NORTH, topPanel);
                windowPane.add(BorderLayout.SOUTH, bottomPanel);

                login.setContentPane(windowPane);
    }

    private static void appFrame() {
        appFrame = new JFrame("Clemson Library");
        appFrame.setMinimumSize(new Dimension(750, 500));
                appFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        appFrame.setLocationRelativeTo(null);

                JPanel windowPanel = appPanel();
                final JPanel windowPane = new JPanel(new BorderLayout());

                windowPane.add(BorderLayout.NORTH, windowPanel);

                appFrame.setContentPane(windowPane);
        }

    private static JPanel appPanel() {
        JPanel upperPanel = new JPanel(new BorderLayout());
                JTabbedPane tabs = tabs();
                JPanel pane = new JPanel(new GridBagLayout());


        String type = "Member";
                if(myLoggedInAccount.getStaff())
                        type = "Staff";
                JComponent user = new JTextArea("User: " + myLoggedInAccount.getMyName() + "\tAccount
Type: " + type);
```

```
                JPanel tools = new JPanel(new FlowLayout(FlowLayout.TRAILING));

                JButton logout = new JButton("Logout");
                logout.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                loggedout = true;
                                myLoggedInAccount = null;
                        }
                });
                JButton exit = new JButton("Exit");
                exit.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                System.exit(0);
                        }
                });
                tools.add(logout);
                tools.add(exit);
                pane.add(user, left);
                pane.add(tools, right);

                upperPanel.add(BorderLayout.NORTH, pane);
                upperPanel.add(BorderLayout.SOUTH, tabs);

                return upperPanel;
    }

    private static JTabbedPane tabs() {
        JTabbedPane tabs = new JTabbedPane();

        JPanel catapanel = CatalogTab();
                tabs.addTab("Catalog", null, catapanel, "The coolest books ever");

                if(myLoggedInAccount.getStaff()) {
                        JPanel accpanel = AccountTab();
                        tabs.addTab("Accounts", null, accpanel, "All of the RADICAL people who like
books");
                }

        JPanel check = Check();
                tabs.addTab("Checkout/Return", null, check, "The place where happiness goes to die");

                return tabs;
    }

    //ALL OF THE CATALOG TAB STUFF
    private static JPanel CatalogTab() {
        JPanel catapanel = new JPanel(new BorderLayout());
        catapanel.setPreferredSize(new Dimension(300, 400));

                catatable = new CatalogTable(myCatalog);
                catatbl = new JTable(catatable);
                catatbl.setPreferredSize(new Dimension(300, 400));
                catatbl.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        JScrollPane scroll = new JScrollPane(catatbl);
        catapanel.add(BorderLayout.CENTER, scroll);


        JPanel left = leftPanelCatalog();
                left.setBorder(new EmptyBorder(50, 0, 50, 0));
        catapanel.add(BorderLayout.WEST, left);

        return catapanel;
    }

    private static JPanel leftPanelCatalog() {
        JPanel panel = new JPanel(new GridLayout(5, 1, 0, 20));
        panel.setSize(75, 400);
        panel.setBorder(new EmptyBorder(100, 10, 5, 10));

        if(myLoggedInAccount.getStaff()) {
                JButton addButton = new JButton("Add Book");
                addButton.setSize(2, 1);
                addButton.setBorder(new EmptyBorder(5, 10, 5, 10));
                addButton.addActionListener(new ActionListener() {
```

```
                @Override
                public void actionPerformed(ActionEvent e) {
                    CatalogDialog dialog = new CatalogDialog(myCatalog.getSize());
                    dialog.initialize();
                    dialog.setVisible(true);
                    if (dialog.wasSuccessful()) {
                        myCatalog.addBook(dialog.getResult());
                        catatable.fireTableDataChanged();
                        try {
                                            save();
                                        } catch (Exception e1) {
                                            //
                                        }
                    }
                }
            });

            JButton removeButton = new JButton("Remove Book");
            removeButton.setSize(2, 1);
            removeButton.setBorder(new EmptyBorder(5, 10, 5, 10));
            removeButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    int index = catatbl.getSelectedRow();
                    if (index >= 0) {
                          myCatalog.removeBook(catatbl.getSelectedRow());
                        catatable.fireTableDataChanged();
                    }
                    try {
                                        save();
                                    } catch (Exception e1) {
                                        //
                                    }
                }
            });

            panel.add(addButton);
            panel.add(removeButton);
        }
        JPanel search = searchPanel();
        panel.add(search);

        return panel;
    }

    private static JPanel searchPanel() {
        JPanel search = new JPanel(new GridBagLayout());
        search.setSize(75, 400);

        JLabel entersearch = new JLabel("Enter search: ");
        final JTextField text = new JTextField();
        final JRadioButton fuzz = new JRadioButton("Fuzzy");
        JRadioButton litral = new JRadioButton("Literal");
        JPanel type = new JPanel(new GridBagLayout());
        type.add(fuzz);
        type.add(litral);

        JPanel buttons = new JPanel();
        JButton srch = new JButton("Search");
        srch.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
               ArrayList<Book> ret;
                if(fuzz.isSelected()) {
                       ret = fuzzy.search(text.getText(), myCatalog.getMyBooks());
                }
                else {
                       ret = literal.search(text.getText(), myCatalog.getMyBooks());
                }
            }
        });
        JButton reset = new JButton("Reset");
        reset.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

            }
```

```
    });
    buttons.add(srch);
    buttons.add(reset);

    ButtonGroup button = new ButtonGroup();
    button.add(fuzz);
    button.add(litral);

    search.add(entersearch, left);
    search.add(text, right);
    search.add(buttons, left);
    search.add(type, right);

    return search;
}
//END CATALOG PANEL STUFF

//ALL OF THE ACCOUNT TAB STUFF
private static JPanel AccountTab() {
    JPanel accpanel = new JPanel(new BorderLayout());
            accpanel.setSize(300, 400);

            JPanel left = leftPanelAccounts();
            left.setBorder(new EmptyBorder(50, 0, 50, 0));

            acctable = new AccountTable(myAccountDatabase);
    acctbl = new JTable(acctable);
    acctbl.setSize(300, 400);
    acctbl.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    JScrollPane scroll = new JScrollPane(acctbl);
    accpanel.add(BorderLayout.CENTER, scroll);
    accpanel.add(BorderLayout.WEST, left);

    return accpanel;
}

private static JPanel leftPanelAccounts() {
    JPanel panel = new JPanel(new GridLayout(8, 1, 0, 20));
    panel.setSize(75, 400);
    panel.setBorder(new EmptyBorder(100, 10, 5, 10));

    JButton addButton = new JButton("Add Account");
    addButton.setSize(2, 1);
    addButton.setBorder(new EmptyBorder(5, 10, 5, 10));
    addButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            AccountDialog dialog = new AccountDialog(myAccountDatabase.getSize());
            dialog.initialize();
            dialog.setVisible(true);
            if (dialog.wasSuccessful()) {
                myAccountDatabase.addAccount(dialog.getResult());
                acctable.fireTableDataChanged();
                try {
                                        save();
                                } catch (Exception e1) {
                                        //
                                }
            }
        }
    });

    JButton removeButton = new JButton("Remove Account");
    removeButton.setSize(2, 1);
    removeButton.setBorder(new EmptyBorder(5, 10, 5, 10));
    removeButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int index = acctbl.getSelectedRow();
            if (index >= 0) {
                  myAccountDatabase.removeAccount(acctbl.getSelectedRow());
                acctable.fireTableDataChanged();
            }
            try {
                                save();
                        } catch (Exception e1) {
                                //
```

```
                        }
        }
    });

    panel.add(BorderLayout.NORTH, addButton);
    panel.add(BorderLayout.SOUTH, removeButton);
    return panel;
}
//END ACCOUNT PANEL STUFF

//CHECKOUT STUFF
private static JPanel Check() {
    JPanel checkpanel = new JPanel(new BorderLayout());
            checkpanel.setSize(750, 400);

            if(!myLoggedInAccount.getStaff()) {
                    JPanel userbooks = userBooks();
                    JPanel actions = actions();
                    JPanel buttons = new JPanel(new GridLayout(8, 1, 0, 20));

                    JButton checkout = new JButton("Checkout");
                    checkout.setBorder(new EmptyBorder(10, 0, 10, 0));
                    checkout.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    JFrame frame = CatalogWindow();
                    frame.setVisible(true);
                }
            });
                    JButton complete = new JButton("Complete");
                    complete.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    cart.completeCheckout();
                    actiontable.fireTableDataChanged();
                    catatable.fireTableDataChanged();
                    catatable = new CatalogTable(myCatalog);
                    catatbl.setModel(catatable);
                    catatable.fireTableDataChanged();
                            usertable = new UserTable(myCatalog.userBooks(myLoggedInAccount));
                            usertbl.setModel(usertable);
                            usertable.fireTableDataChanged();
                    try {
                                        save();
                                } catch (Exception e1) {
                                        //
                                }
                }
            });
                    complete.setBorder(new EmptyBorder(10, 0, 10, 0));
                    buttons.add(checkout);
                    buttons.add(complete);

                    checkpanel.add(BorderLayout.WEST, userbooks);
                    checkpanel.add(BorderLayout.CENTER, buttons);
                    checkpanel.add(BorderLayout.EAST, actions);
            }
            else {
                    JPanel err = new JPanel(new GridLayout(3, 1, 0, 0));
                    JLabel error1 = new JLabel("ERROR YOU CAN'T CHECKOUT BOOKS");
                    JLabel error2 = new JLabel("BECAUSE YOU'RE STAFF AND");
                    JLabel error3 = new JLabel("CAN'T HAVE FUN");
                    error1.setFont(new Font("Serif", Font.BOLD, 30));
                    error2.setFont(new Font("Serif", Font.BOLD, 30));
                    error3.setFont(new Font("Serif", Font.BOLD, 30));

                    err.add(error1);
                    err.add(error2);
                    err.add(error3);
                    checkpanel.add(BorderLayout.CENTER, err);
            }

            return checkpanel;
}

private static JPanel userBooks() {
    JPanel userpanel = new JPanel(new BorderLayout());
```

```
    JLabel header = new JLabel("Books you have checked out:");
    usertable = new UserTable(myCatalog.userBooks(myLoggedInAccount));
            usertbl = new JTable(usertable);
            usertbl.addMouseListener(new MouseAdapter() {
                public void mousePressed(MouseEvent me) {
                    JTable table =(JTable) me.getSource();
                    Point p = me.getPoint();
                    int row = table.rowAtPoint(p);
                    if (me.getClickCount() == 2) {
                        cart.returnBook(usertable.getBookAt(row), myLoggedInAccount);
                        actiontable.fireTableDataChanged();
                    }
                }
            });
            usertbl.setPreferredSize(new Dimension(300, 400));
            usertbl.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            usertbl.setPreferredScrollableViewportSize(usertbl.getPreferredSize());
            usertbl.setFillsViewportHeight(true);
     JScrollPane scroll = new JScrollPane(usertbl);

     userpanel.add(BorderLayout.NORTH, header);
     userpanel.add(BorderLayout.WEST, scroll);

     return userpanel;
}

private static JPanel actions() {
    JPanel userpanel = new JPanel(new BorderLayout());

    JLabel header = new JLabel("Actions to complete:");

    actiontable = new ActionTable(cart);
            actiontbl = new JTable(actiontable);
            actiontbl.setPreferredSize(new Dimension(300, 400));
            actiontbl.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            actiontbl.setPreferredScrollableViewportSize(actiontbl.getPreferredSize());
            actiontbl.setFillsViewportHeight(true);
     JScrollPane scroll = new JScrollPane(actiontbl);

     userpanel.add(BorderLayout.NORTH, header);
     userpanel.add(BorderLayout.EAST, scroll);

     return userpanel;
}

public static JFrame CatalogWindow() {
    final JFrame frame = new JFrame("Catalog");
    frame.setPreferredSize(new Dimension(400, 400));
    frame.setMinimumSize(new Dimension(400, 400));
    frame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    frame.setLocationRelativeTo(null);

    JPanel catapanel = new JPanel(new BorderLayout());
    final JTable tempcata = catatbl;
    JButton select = new JButton("Select");
    select.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int index = tempcata.getSelectedRow();
           cart.checkoutBook(myCatalog.getBook(index), myLoggedInAccount);
           actiontable.fireTableDataChanged();
           frame.dispatchEvent(new WindowEvent(frame, WindowEvent.WINDOW_CLOSING));
        }
     });
    JScrollPane scroll = new JScrollPane(catatbl);
     catapanel.add(BorderLayout.CENTER, scroll);
    catapanel.add(BorderLayout.EAST, select);

    JPanel windowPanel = catapanel;
            final JPanel windowPane = new JPanel(new BorderLayout());

            windowPane.add(BorderLayout.NORTH, windowPanel);

            frame.setContentPane(windowPane);
            return frame;
}
```

```
    //END CHECKOUT STUFF

    private static void save() throws FileNotFoundException, IOException {
        File datFile = new File("input.dat");
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(datFile));
        oos.writeObject(myAccountDatabase);
        oos.writeObject(myCatalog);
        oos.close();
    }

    private static void load() throws FileNotFoundException, IOException, ClassNotFoundException {
        File datFile = new File("input.dat");
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(datFile));
        myAccountDatabase = (AccountDatabase) ois.readObject();
        myCatalog = (Catalog) ois.readObject();
        ois.close();
    }
}
```

AccountTable.java

```java
package cu.cs.cpsc2150.project3;

import javax.swing.table.AbstractTableModel;

import cu.cs.cpsc2150.project2.Account;
import cu.cs.cpsc2150.project2.AccountDatabase;
import cu.cs.cpsc2150.project2.Book;
import cu.cs.cpsc2150.project2.Catalog;

public class AccountTable extends AbstractTableModel{
        private static final String[] columnHeaders = { "ID", "Name", "Username", "Password", "Type"
};

        private AccountDatabase myData;

    public AccountTable(AccountDatabase data) {
        myData = data;
    }

    @Override
    public int getRowCount() {
        return myData.getSize();
    }

    @Override
    public int getColumnCount() {
        return columnHeaders.length;
    }

    @Override
    public String getColumnName(int column) {
        return columnHeaders[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Account thing = myData.getAccount(rowIndex);
        switch (columnIndex) {
            case 0:
                return thing.getID();
            case 1:
                return thing.getMyName();
            case 2:
                return thing.getUsername();
            case 3:
                return thing.getPassword();
            case 4:
                return thing.getType();
            default:
                return "???";
        }
    }
}
```

AccountDialog.java

```
package cu.cs.cpsc2150.project3;
import javax.swing.*;
import javax.swing.border.EmptyBorder;

import cu.cs.cpsc2150.project2.Account;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;

public class AccountDialog extends JDialog {
    private Account myDataThing = null;
    private boolean mySuccessFlag;
    private int id;

    public AccountDialog(int id) {
        super(new JFrame(), "New Account", true);
        this.setSize(new Dimension(300, 300));
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(HIDE_ON_CLOSE);
        mySuccessFlag = false;
        this.id = id;
    }

    public void initialize() {
        JPanel windowPanel = new JPanel(new GridBagLayout());
               GridBagConstraints left = new GridBagConstraints();
        left.anchor = GridBagConstraints.LINE_START;
        GridBagConstraints right = new GridBagConstraints();
               right.weightx = 2.0;
        right.fill = GridBagConstraints.HORIZONTAL;
        right.gridwidth = GridBagConstraints.REMAINDER;
        windowPanel.setBorder(new EmptyBorder(10, 10, 10, 10));

        JLabel name = new JLabel("Name:");
        name.setBorder(new EmptyBorder(2, 2, 2, 2));
        JLabel username = new JLabel("Username:");
        username.setBorder(new EmptyBorder(2, 2, 2, 2));
        JLabel password = new JLabel("Password:");
        password.setBorder(new EmptyBorder(2, 2, 2, 2));
        JLabel type = new JLabel("Account Type:");
        JLabel email = new JLabel("Email:");
        JLabel phone = new JLabel("Phone Number: ");
        JLabel phoneEx = new JLabel("Ex: XXX-XXX-XXXX");
        phoneEx.setSize(12, 1);
        final JLabel invalEmail = new JLabel("Not a valid email!");
        invalEmail.setVisible(false);
        invalEmail.setForeground(Color.red);
        final JLabel invalPhone = new JLabel("Not a valid phone number!");
        invalPhone.setVisible(false);
        invalPhone.setForeground(Color.red);
        type.setBorder(new EmptyBorder(2, 2, 2, 2));
        final JRadioButton staf = new JRadioButton("Staff");
        JRadioButton mem = new JRadioButton("Member");
        ButtonGroup group = new ButtonGroup();
        group.add(staf);
        group.add(mem);
        final JTextField nameField = new JTextField();
        final JTextField usernameField = new JTextField();
        final JPasswordField passwordField = new JPasswordField();
        final JTextField emailField = new JTextField();
        emailField.addFocusListener(new FocusAdapter() {
            public void focusLost(FocusEvent e) {
                if(!(emailField.getText()).matches("[a-zA-Z0-9]+@[a-zA-Z0-9]+\\.[a-zA-z]{2,}"))
                       invalEmail.setVisible(true);
            }
            public void focusGained(FocusEvent e) {
               invalEmail.setVisible(false);
            }
        });
        final JTextField phoneField = new JTextField();
        phoneField.addFocusListener(new FocusAdapter() {
```

```
            public void focusLost(FocusEvent e) {
                if(!(phoneField.getText()).matches("^(\\+\\d{1,2}\\s)?\\(?\\d{3}\\)?[\\s.-]\\d{3}
[\\s.-]\\d{4}$"))
                        invalPhone.setVisible(true);
            }
            public void focusGained(FocusEvent e) {
               invalPhone.setVisible(false);
            }
        });

        JButton okButton = new JButton("OK");
        okButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
               if(!invalEmail.isVisible() && !invalPhone.isVisible()) {
                        try {
                              boolean isstaff = false;
                              if(staf.isSelected())
                                      isstaff = true;
                            myDataThing = new Account(id, nameField.getText(),
usernameField.getText(),String.valueOf(passwordField.getPassword()),
                                                                                    emailField.getText(),
phoneField.getText(), isstaff);
                              mySuccessFlag = true;
                              AccountDialog.this.setVisible(false);
                         } catch (NumberFormatException ex) {
                              JOptionPane.showMessageDialog(AccountDialog.this, "Invalid account.",
"Error", JOptionPane.ERROR_MESSAGE);
                         }
                }
                else {
                        JOptionPane.showMessageDialog(AccountDialog.this, "Invalid account.",
"Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                AccountDialog.this.setVisible(false);
            }
        });

        JPanel butt = new JPanel(new FlowLayout());
        butt.add(staf);
        butt.add(mem);

        windowPanel.add(name, left);
        windowPanel.add(nameField, right);
        windowPanel.add(username, left);
        windowPanel.add(usernameField, right);
        windowPanel.add(password, left);
        windowPanel.add(passwordField, right);
        windowPanel.add(type, left);
        windowPanel.add(butt, right);
        windowPanel.add(email, left);
        windowPanel.add(emailField, right);
        windowPanel.add(invalEmail, right);
        windowPanel.add(phone, left);
        windowPanel.add(phoneField, right);
        windowPanel.add(phoneEx, right);
        windowPanel.add(invalPhone, right);
        windowPanel.add(okButton, left);
        windowPanel.add(cancelButton, right);

        this.getContentPane().add(windowPanel);
    }

    public boolean wasSuccessful() {
        return mySuccessFlag;
    }

    public Account getResult() {
        return myDataThing;
    }
```

```
}
```

CatalogTable.java

```java
package cu.cs.cpsc2150.project3;
import javax.swing.table.AbstractTableModel;

import cu.cs.cpsc2150.project2.Book;
import cu.cs.cpsc2150.project2.Catalog;

// the table model supplies the data that will appear in a JTable
public class CatalogTable extends AbstractTableModel {
    private static final String[] columnHeaders = { "ID", "Title", "Author", "Genre", "Checked
Out" };

    private Catalog myData;

    public CatalogTable(Catalog data) {
        myData = data;
    }

    @Override
    public int getRowCount() {
        return myData.getSize();
    }

    @Override
    public int getColumnCount() {
        return columnHeaders.length;
    }

    @Override
    public String getColumnName(int column) {
        return columnHeaders[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Book thing = myData.getBook(rowIndex);
        switch (columnIndex) {
            case 0:
                return thing.getID();
            case 1:
                return thing.getMyTitle();
            case 2:
                return thing.getMyAuthor();
            case 3:
                return thing.getMyGenre();
            case 4:
                return thing.checkedout();
            default:
                return "???";
        }
    }
}
```

CatalogDialog.java

```
package cu.cs.cpsc2150.project3;
import javax.swing.*;
import javax.swing.border.EmptyBorder;

import cu.cs.cpsc2150.project2.Book;
import cu.cs.cpsc2150.project2.Catalog;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.StringTokenizer;

public class CatalogDialog extends JDialog {
    private Book myDataThing = null;
    private boolean mySuccessFlag;
    private int id;
    private JRadioButton rom, com, scifi, horror, action, psych, thrill, fantasy;
    private ButtonGroup group;
    private GridBagConstraints left, right;

    public CatalogDialog(int id) {
        super(new JFrame(), "New Book", true);
        this.setSize(new Dimension(400, 250));
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(HIDE_ON_CLOSE);
        mySuccessFlag = false;
        this.id = id;
    }

    public void initialize() {
        JPanel windowPanel = new JPanel(new GridBagLayout());
        left = new GridBagConstraints();
        left.anchor = GridBagConstraints.LINE_START;
        right = new GridBagConstraints();
               right.weightx = 2.0;
        right.fill = GridBagConstraints.HORIZONTAL;
        right.gridwidth = GridBagConstraints.REMAINDER;
        windowPanel.setBorder(new EmptyBorder(10, 10, 10, 10));

        JLabel title = new JLabel("Title:");
        title.setBorder(new EmptyBorder(2, 2, 2, 2));
        JLabel author = new JLabel("Author:");
        author.setBorder(new EmptyBorder(2, 2, 2, 2));
        JLabel genre = new JLabel("Genre:");
        genre.setBorder(new EmptyBorder(2, 2, 2, 2));
        JPanel buttons = genreButtons();
        JLabel tags = new JLabel("Tags:");
        tags.setBorder(new EmptyBorder(2, 2, 2, 2));

        final JTextField titleField = new JTextField();
        final JTextField authorField = new JTextField();
        final JTextField tagsField = new JTextField();

        JButton okButton = new JButton("OK");
        okButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                       StringTokenizer token = new StringTokenizer(tagsField.getText(), ",");
                       ArrayList<String> arr = new ArrayList<String>();
                       while(token.hasMoreTokens()) {
                              arr.add(token.nextToken());
                       }
                       String gen = findGenre();
                    myDataThing = new Book(id, titleField.getText(), authorField.getText(), gen,
arr);
                    mySuccessFlag = true;
                    CatalogDialog.this.setVisible(false);
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(CatalogDialog.this, "Invalid number.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
```

```
        });

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                CatalogDialog.this.setVisible(false);
            }
        });

        windowPanel.add(title, left);
        windowPanel.add(titleField, right);
        windowPanel.add(author, left);
        windowPanel.add(authorField, right);
        windowPanel.add(genre, left);
        windowPanel.add(buttons, right);
        windowPanel.add(tags, left);
        windowPanel.add(tagsField, right);
        windowPanel.add(okButton);
        windowPanel.add(cancelButton);

        this.getContentPane().add(windowPanel);
    }

    public boolean wasSuccessful() {
        return mySuccessFlag;
    }

    public Book getResult() {
        return myDataThing;
    }

    private JPanel genreButtons() {
        JPanel panel = new JPanel(new GridBagLayout());
        rom = new JRadioButton("Romance");
        com = new JRadioButton("Comedy");
        scifi = new JRadioButton("Science Fiction");
        horror = new JRadioButton("Horror");
        action = new JRadioButton("Action");
        psych = new JRadioButton("Psychological");
        thrill = new JRadioButton("Thriller");
        fantasy = new JRadioButton("Fantasy");

        group = new ButtonGroup();
        group.add(rom);
        group.add(com);
        group.add(scifi);
        group.add(horror);
        group.add(action);
        group.add(psych);
        group.add(thrill);
        group.add(fantasy);

        panel.add(rom, left);
        panel.add(com, right);
        panel.add(scifi, left);
        panel.add(horror, right);
        panel.add(action, left);
        panel.add(psych, right);
        panel.add(thrill, left);
        panel.add(fantasy, right);

        return panel;
    }

    private String findGenre() {
        if(rom.isSelected())
                return "Romance";
        else if(com.isSelected())
                return "Comedy";
        else if (scifi.isSelected())
                return "Science Fiction";
        else if (horror.isSelected())
                return "Horror";
        else if(action.isSelected())
                return "Action";
        else if (psych.isSelected())
```

```
                return "Psychological";
        else if (thrill.isSelected())
                return "Thrill";
        else
                return "Fantasy";
    }
}
```

ActionTable.java

```java
package cu.cs.cpsc2150.project3;

import java.util.ArrayList;

import javax.swing.table.AbstractTableModel;

import cu.cs.cpsc2150.project2.Book;
import cu.cs.cpsc2150.project2.Cart;
import cu.cs.cpsc2150.project2.CartItem;

public class ActionTable extends AbstractTableModel {
        private static final String[] columnHeaders = { "Action", "Book" };

    private Cart myData;

    public ActionTable(Cart data) {
        myData = data;
    }

    @Override
    public int getRowCount() {
        return myData.getSize();
    }

    @Override
    public int getColumnCount() {
        return columnHeaders.length;
    }

    @Override
    public String getColumnName(int column) {
        return columnHeaders[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        CartItem thing = myData.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return thing.getCommand();
            case 1:
                return thing.getBook().getMyTitle();
            default:
                return "???";
        }
    }

    public CartItem getItemAt(int ndx) {
       return myData.get(ndx);
    }
}
```

UserTable.java

```java
package cu.cs.cpsc2150.project3;

import java.util.ArrayList;

import javax.swing.table.AbstractTableModel;

import cu.cs.cpsc2150.project2.Book;
import cu.cs.cpsc2150.project2.Catalog;

public class UserTable extends AbstractTableModel {
        private static final String[] columnHeaders = { "Title", "Author", "Genre" };

    private ArrayList<Book> myData;

    public UserTable(ArrayList<Book> data) {
        myData = data;
    }

    @Override
    public int getRowCount() {
        return myData.size();
    }

    @Override
    public int getColumnCount() {
        return columnHeaders.length;
    }

    @Override
    public String getColumnName(int column) {
        return columnHeaders[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Book thing = myData.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return thing.getMyTitle();
            case 1:
                return thing.getMyAuthor();
            case 2:
                return thing.getMyGenre();
            default:
                return "???";
        }
    }

    public Book getBookAt(int ndx) {
       return myData.get(ndx);
    }
}
```

CatalogWindow.java

```java
package cu.cs.cpsc2150.project3;

import java.awt.Dimension;

import javax.swing.JFrame;

public class CatalogWindow extends JFrame {
        public CatalogWindow() {
        super("Catalog");
        this.setPreferredSize(new Dimension(400, 400));
        this.setMinimumSize(new Dimension(750, 500));
        this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        this.setLocationRelativeTo(null);
    }
}
```