**Thomas Ortiz - 200052864**

**Corey Colberg - 200047978**

You are able to run the program with either including a argument when launching the GUI - WolfLibraryGUI.java - or you can launch the GUI and preceed with what it asks you. The file you launch with or pick should contain a list of books that will become the WolfLibrary.

Inventory

* **Book** - This class creates a library book .
* **BookDB** - BookDB represents an internal database of books - (The books from the file you picked).

Lending\_System

* **LendingManager** - Interface for a lending system where the available items are stored in an inventory and where there are different users. Items can be reserved, checked out for home, and returned to the inventory. Items in the the inventory, reserves, and at home can be located by position.
* **LibraryLendingSystem** - Describes the library system operations.

Patron

* **AccountManager** - Describes behaviors of a patron management systems that permits user login. The management system has an administrator.
* **Admin** - Creates an Administrator for the Wolf Library.
* **LibraryAccountSystem.java** - Describes the patron management part of the Wolf Library.

Noticed a few problems here with logout.

* **Patron** - This class is responsible for making the patron object. Each patron consists of a username, password, max number of books he/she can check out, a list of books he/she has checked out, and a list of books the he/she has on reserve queue. The public methods traverseReserveQueue and traverseCheckedOut are return a string of the books in each respective list, seperated by a new line. The closeAccount method closes a patron's account by returning all of his books that are checked out. To return a patron's book, the returnBook method is used which returns the book at a given index. MoveAheadOneInReserve moves the book at the given index and moves it up one spot on the reserveQueue. If the index is zero nothing happens. UnReserve takes a book of the reserveQueue list. Reserve adds a book the the reserve queue and if the user can check out more books it will check out the first book on the reserveQueue.Three private methods are used in Patron. traverseQueue takes a list as a parameter and goes through it returning each book on a new line of a string. CheckOut checks a book out if there is room on the checkedOut list. The last private method, removeFirstAvailable, removes first available book from reserveQueue. Two additional methods were added for testing purposes, getReserveQueue and getCheckedOut, which are getter methods for the two list.
* **PatronDB** - This class is a database of the patrons in the system. It consists of an ArrayList of patrons. The method verifyPatron takes a username and password and checks to see if the Patron exist in the database. If it does it will return the Patron. The method listAccounts simply returns a string of all the patron accounts (used mainly for testing) separated by a new line. To add a new Patron to the database the addNewPatron method is used which creates a new Patron then adds it to the ArrayList. CancelAccount finds the patron account with the username and deletes the account from the database and returns all the patrons books. Three private methods isNewPatron, insert, and findMatchingPatron are used in the public methods above. IsNewPatron returns a bool value on whether the account username already exist (useful for new accounts, and checking to see if you can delete an account). Insert simply adds a Patron object to end of the ArrayList<Patron>. FindMatchingPatron returns the patron index with the same username as the parameter.
* **User** - Represents a User of the Wolf Library.

Util

* **Constants** - Contains constants for the WolfLibrary project. These constants can be used in the program and for testing.
* **List** - A generic class that supports the underlying list operations for the library inventory, reserve queues, and checked out queues.
  + \* DISCLAIMER\*\*\*: We were told we could use ArrayLists instead of MultiPurposeLists for this class. As such, we tried out best to keep MultiPurposeList as close to what it was.