CS 1073

FR04B

Assignment 12

Ebrahim Arefi

3621326

A. Writing Three Classes

ResidentMember.java

```
/**
 * Represents a resident's membership card that allows them to sign out items.
 * It holds information about their
 * name, room number, chat name and membership ID.
 *
 * @author Ebrahim Arefi
 */
public class ResidentMember {

    private String name;
    private int roomNumber;
    private final String chatName;
    private final int memberId;
    private static int nextId = 100000;

    private LibraryItem[] signedOut;
    private int count;

    /**
     * Constructor method:
     *
     * @param name     Full name of the resident.
     * @param roomNum  Room number of the resident.
     * @param chatName Chat username of the resident.
     */
    public ResidentMember(String name, int roomNum, String chatName) {
        this.name = name;
        this.roomNumber = roomNum;
        this.chatName = chatName;

        memberId = nextId++;
        signedOut = new LibraryItem[10];
        count = 0;
    }

    /**
     * Accessor method: Returns full name.
     *
     * @return Full name of the resident.
     */
    public String getName() {
        return name;
    }
```

```
/**
 * Accessor method: Returns the room number.
 *
 * @return Room number.
 */
public int getRoomNumber() {
    return roomNumber;
}

/**
 * Accessor method: Returns the chat username.
 *
 * @return Chat name.
 */
public String getChatName() {
    return chatName;
}

/**
 * Accessor method: Returns membership ID.
 *
 * @return Membership ID.
 */
public int getMembershipNumber() {
    return memberId;
}

/**
 * Mutator method: Changes the name of the member.
 *
 * @param nameIn New name of the resident.
 */
public void setName(String nameIn) {
    name = nameIn;
}

/**
 * Mutator method: Changes the room number of the resident.
 *
 * @param roomNumIn New room number of the resident.
 */
public void setRoomNumber(int roomNumIn) {
    roomNumber = roomNumIn;
}

/**
 * Accessor method: Copies the array for the library items.
 *
 * @return A copy of the array containing only the valid items.
 */
public LibraryItem[] getSignedOutItems() {
    LibraryItem[] copyItems = new LibraryItem[count];
```

```
        for (int i = 0; i < count; i++) {
            copyItems[i] = signedOut[i];
        }

        return copyItems;
    }

    /**
     * Mutator method: Signs out a library item.
     *
     * @param outItem Library item to be signed out.
     * @return True if the item was successfully signed out.
     */
    public boolean signOut(LibraryItem outItem) {

        if (count >= 10) {
            return false;
        }

        signedOut[count] = outItem;
        count++;
        return true;
    }

    /**
     * Mutator method: Returns a library item previously signed out
     *
     * @param returnedItem The library item to be returned.
     * @return True if the item was successfully returned, false if not.
     */
    public boolean returnItem(LibraryItem returnedItem) {

        for (int i = 0; i < count; i++) {
            if (signedOut[i].equals(returnedItem)) {

                for (int j = i; j < count - 1; j++) {
                    signedOut[j] = signedOut[j + 1];
                }

                signedOut[count - 1] = null;
                count--;
                return true;
            }
        }

        return false;
    }
}
```

LibraryItem.java

```
/**
 * Represents library item's description.
 * A library item may be signed out by resident members unless its restricted.
 *
 * @author Ebrahim Arefi, 3621326
 */
public class LibraryItem {
    private final String description;
    private final double price;
    private final boolean isDonatedByAlumni;

    /**
     * Constructor method:
     *
     * @param descriptionIn     Description of the library item.
     * @param priceIn           Purchase price of the library item.
     * @param donatedByAlumniIn True if the Alumni funded the item.
     */
    public LibraryItem(String descriptionIn, double priceIn, boolean donatedByAlumni) {

        description = descriptionIn;
        price = priceIn;
        isDonatedByAlumni = donatedByAlumni;
    }

    /**
     * Accessor method:
     * Returns description of the item.
     *
     * @return
     */
    public String getDescription() {
        return description;
    }

    /**
     * Accessor method:
     * Returns the price of the item.
     *
     * @return
     */
    public double getPrice() {
        return price;
    }

    /**
     * Accessor method:
     * Checks if the library item was funded by the Alumni Association.
     *
     * @return
     */
    public boolean getAlumniDonated() {
        return isDonatedByAlumni;
    }
}
```

StaffResidentMember.java

```
/**
 * Is a child class of the ResidentMember class.
 * Represents a staff member.
 * Holds information of the super class, and members.
 *
 * @author Ebrahim Arefi
 */
public class StaffResidentMember extends ResidentMember {

    private String jobTitle;

    /**
     * Constructor method:
     *
     * @param nameIn     Full name of the staff.
     * @param roomNumIn  Room number of the staff.
     * @param chatNameIn Chat username of the staff.
     * @param jobTitleIn Job title of the staff.
     */
    public StaffResidentMember(String nameIn, int roomNumIn, String chatNameIn, String
jobTitleIn) {
        super(nameIn, roomNumIn, chatNameIn);
        jobTitle = jobTitleIn;
    }

    /**
     * Accessor method:
     * Returns the job title of the staff.
     *
     * @return
     */
    public String getJobTitle() {
        return jobTitle;
    }

    /**
     * Mutator method:
     * Staff cannot sign out items that were donated by alumni.
     *
     * @param wantedItem The library item to be signed out.
     * @return True if the item is not funded by alumni, else false.
     */
    public boolean signOut(LibraryItem wantedItem) {

        if (wantedItem.getAlumniDonated()) {
            return false;
        }

        return super.signOut(wantedItem);
    }
}
```

OutPut:

```
*** Test case #1: Create a ResidentMember object & test accessors
Name:     Maria Diaz
Unit #:   163
Phone:    mdiaz03
Member #: 100000
Correct result: Maria has zero lending items.

*** Test case #2: Create a StaffResidentMember object & test accessors
Name:     Rory MacDonald
Unit #:   306
Phone:    rmacdo12
Member #: 100001
Departs:  Music Teacher
Correct result: Rory has zero lending items.

*** Test case #3: Automatically generate a member number
Correct result: 100002 is the correct member number.

*** Test case #4: Create a LibraryItem object & test accessors
Description:    Skip-Bo (Card Game)
Original Price: $11.25
Alumni Donated: true

*** Test case #5: Change name for StaffResidentMember
Correct result: Rory's name successfully changed.

*** Test case #6: Change room number for both resident types
Correct result: Maria's room number successfully changed.
Correct result: Rory's room number successfully changed.

*** Test case #7: Sign out one LibraryItem
Correct result: Maria signed out an item successfully.
Correct result: Maria has one lending item.

*** Test case #8: Sign out multiple LibraryItems
Correct result: Maria signed out two more items successfully.
Correct result: Maria has three lending items.

*** Test case #9: Intentionally exceed the sign out limit
Correct result: Maria was prevented from signing out more than 10 lending
items.

*** Test case #10: A staff resident tries to sign out items
```

```
Correct result: Rory was prevented from signing out an alumni-donated item.
Correct result: Rory was able to sign out a non-alumni-donated item.

*** Test case #11: Returning the only item that was signed out
Correct result: Rory's item was successfully returned.
Correct result: Rory's list length changed appropriately.

*** Test case #12: Returning an item that was not signed out
Correct result: Unsuccessful attempt to return an item that was not signed
out.

*** Test case #13: Returning the first item that was signed out
Correct result: Maria's first item was successfully returned.
Correct result: Maria's list length changed appropriately.

Confirm return: Skip-Bo should be absent from the following list:
Connect 4 (Board Game)
Frisbee
Cribbage Board and Cards
Codenames (Card Game)
Badminton Raquets (2) and Birdies
Ladder Ball Set
Scrabble (Board Game)
Spikeball Game Set
Karaoke Machine

*** Test case #14: Returning a mid-list item
Correct result: Codenames (Card Game) was successfully returned.
Correct result: Maria's list length changed appropriately.

Confirm return: Codenames (Card Game) should be absent from the following
list:
Connect 4 (Board Game)
Frisbee
Cribbage Board and Cards
Badminton Raquets (2) and Birdies
Ladder Ball Set
Scrabble (Board Game)
Spikeball Game Set
Karaoke Machine

********** End of Test Cases **********
```