CSE214

HOMEWORK - SPRING 2017

HOMEWORK 1 - due Tuesday, February 7th no later than 6:00pm

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework assignments MUST be entirely your own work. Also, group efforts are not allowed.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You are not allowed to use ArrayList, Vector or any other Java API Data Structure classes to implement this assignment except where noted.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

Stony Brook University has recently opened some new residence halls. Since these halls are much more modern than the other residence halls, the university would like to keep track of the students who live there electronically. You will write a program that will manage the students that live in a residence hall for the residence hall director. Your program will model a residence hall with three floors, each of which will be modeled by an array, containing student objects. For each student we will need to keep track of their name, id, and the number of times the student has been written up (documented). If a student has been written up 3 times, they will be kicked out of the residence hall. To make the RA's job easier, all the students on the floor will be placed in the lowest available room number (there will be no holes in the array). If a student moves out, all the students remaining must be shifted down one room in order to make them closer to the the RA at the beginning of the hallway. Also, sometimes, maintenance work must be done on one floor, so therefore students must be moved to another floor. For the convenience of the building administrator (RHD), you will also have to implement a deep copy (clone) function, so the RHD can experiment with alternative floor plans without ruining their original, genius floor plan.

NOTE: All exceptions explicitly thrown in Required Classes except for IllegalArgumentException are custom exceptions that need to be made by you.

Required Classes

Residence Hall Manager (Driver Class)

- public static void main(String[] args)
  - Runs the User interface menu as shown in Required Functions
  - Can be changed as necessary for GUI/Android App
  - This method will contain **three** Floor objects for cloning. You may use an array of Floor objects if you like, but this is not required. We will refer to them as Floor 1, Floor 2, and Floor 3

Floor //If you're familiar with ArrayList, this works in a somewhat similar way
Write a fully documented class named Floor which stores a list of Students in an array and provides an interface to interact with this list. A Floor can contain up to 50 students at a time, so use the final variable CAPACITY = 50. The class will be based on the following ADT specification:

- private Student[] students
- Final int CAPACITY = 50 /* If this number is changed, your code should work with the
*new number. The number 50 should not be present elsewhere in your code, you should
*instead reference CAPACITY */
- public Floor()
  - Constructor: initializes students to a new Student array of size CAPACITY.
  - *Postcondition*: This Floor has been initialized to an empty list of Students.
- public void addStudent(Student student, int position)
  - adds the student at the desired position. if there are students at/after that position, they will be shifted to the right
  - *Preconditions*: This Floor object has been instantiated and 0 <= position <= items_currently_in_list or 1<= position <=items_currently_in_list + 1 depending on your implementation. The number of Student objects in this Floor is less than CAPACITY.
  - *Postconditions*: The new Student is now stored at the desired position in the Floor. All Students were originally in positions greater than or equal to position are moved back one position. (Ex: if there are 5 Students in a Floor, positions 1-5, and you insert a new Student at position 4, the new Student will be at position 4, the Student that was at position 4 will be moved to position 5, and the Student that was at position 5 will be moved to position 6).
  - **Throws:**
    - FullFloorException if the floor is full
    - IllegalArgumentException if the index entered is invalid (holes left in the array, negative)
- public Student removeStudent(int position)

- ○ Removes and returns the student from the given position. All subsequent students are shifted left to remove the hole in the array
- ○ *Precondition*: This Floor object has been instantiated and 0 <= position <= items_currently_in_list.
- ○ *Postcondition*: The Student at the desired position in the Floor has been removed. All Students that were originally in positions greater than or equal to position are moved forward one position. (Ex: If there are 5 items in a Floor, positions 1-5, and you remove the item at position 4, the item that was at position 5 will be moved to position 4).
- ○ **Throws:**
    - ■ EmptyFloorException if the floor is empty
    - ■ IllegalArgumentException if the index given is too high (ie: no student at that index) or if the index given is negative

- ● public Student getStudent(int position)
    - ○ returns the Student at the given position
    - ○ *Precondition*: This Floor object has been instantiated and 1 <= position <= items_currently_in_list, or 0<= position<=items_currently_in_list -1 depending on your implementation
    - ○ **Throws:**
        - ■ IllegalArgumentException if there is no student at that position (ie: too high, or negative)
- ● public void setStudent(Student student, int position)
    - ○ sets the spot at that position to the given student
    - ○ **Throws:**
        - ■ IllegalArgumentException if the position cannot be set (ie: too high, or negative)

- ● public int count()
    - ○ Returns the number of students in the floor
    - ○ **This should run in O(1) time.**
- ● public Floor clone()
    - ○ Makes a deep copy of the floor, and all the students.

Student //You can use your IDE to automatically generate getters and setters, and constructors
Write a fully documented class named Student which contains the Student's name (String), ID number (int) and number of writeups (int). You should provide accessor and mutator methods for each variable, as well as a constructor for the class.

- ● Final int MAX_WRITEUPS = 3
- ● private String name
- ● private int idNumber
- ● private int numWriteups
- ● Getters and Setters for all of the above variables
- ● public Student clone() //Makes cloning a floor much more elegant

Exceptions //Create an empty class that simply extends Exception

- ● Full Floor Exception
- ● Empty Floor Exception

General Recommendations

You might want to implement a toString() method for Student and Floor to make debugging and printing easier. You do not have to do this, but it will help you.

You can feel free to add extra methods and variables if you need.

UI Required Functions

The menu should have options similar to these, and NOT be case sensitive.
Note: the user should see rooms numbered 1 to CAPACITY. If you choose to make the array size CAPACITY+1 and leave position 0 blank, that's fine. If you want to subtract one from every index the user enters (and add one to indices for display) that's ok too.
The menu should not be case sensitive.

- A - Add Student
  - Name
  - ID
  - Position
- R - Remove Student
  - Position
- S - Swap Students
  - Floor and Position for both
- M - Move Student
  - Original floor and position, destination floor and position
  - Note: if the student cannot be moved, they should remain where they were originally
- F - Select floor
  - Floor number
  - Floor 1 should be selected when the program loads
- C - Copy Floor
  - Source Floor Number
  - Destination floor number
  - This should be a deep copy
- P - Print Current Floor
  - Please try to make it neat (use printf if you can)
- W - Write Up Student
  - Student name - must match exactly
  - If the student has reached MAX_WRITEUPS, they should be removed from the floor
  - It should only search the current floor. There will not be duplicate names on the same floor.
- Q - Quit.

Program Sample

```
Welcome to RockStar Rez, the second worst housing management System at SBU.

Menu:
    A) Add a student
```

```
        R) Remove a student
        S) Swap Students
        M) Move Student
        F) Select Floor
        C) Copy Floor
        P) Print Current Floor
        W) Write Up Student
        Q) Quit

Please select an option: A
Add A Student:
Please enter a name: Samuel Stanley
Please enter an id number: 1
Please enter a spot number: 1

Samuel Stanley added to Floor 1 Room 1.

Please select an option: A
Add A Student:
Please enter a name: Ahmad Esmaili
Please enter an id number: 214
Please enter a spot number: 2

Ahmad Esmaili added to Floor 1 Room 2.

Please select an option: A
Add A Student:
Please enter a name: Paul Fodor
Please enter an id number: 114
Please enter a spot number: 2

Paul Fodor added to Floor 1 Room 2.

Please select an option: P

Floor 1:
Room      Name            ID   Writeups
-----------------------------------------------------------
1      Samuel Stanley     1      0
2      Paul Fodor          114      0
3      Ahmad Esmaili           214      0

Please select an option: A
Add A Student:
Please enter a name: Ritwik Banerjee
Please enter an id number: 214
Please enter a spot number: 5

Invalid Spot Number.
```

```
Please select an option: A
Add A Student:
Please enter a name: Ritwik Banerjee
Please enter an id number: 219
Please enter a spot number: 2

Ritwik Banerjee added to Floor 1 Room 2.

Please select an option: P

Floor 1:
Room     Name           ID   Writeups
------------------------------------------------------------
1     Samuel Stanley     1      0
2     Ritwik Banerjee    219     0
3     Paul Fodor         114     0
4     Ahmad Esmaili          214     0

Please select an option: M

Please enter the source floor: 1
Please enter the source room: 2
Please enter the destination floor: 2
Please enter the destination room: 1

Ritwik Banerjee moved to Floor 2 room 1.

Please select an option: F

Please select a floor: 2

Floor 2 selected.

Please select an option: P

Floor 2:
Room     Name           ID   Writeups
------------------------------------------------------------
1     Ritwik Banerjee    219     0


Please select an option: F

Please select a floor: 1

Floor 1 selected.

Please select an option: S
Please enter the Student 1 floor: 1
Please enter the Student 1 room: 1
Please enter the Student 2 floor: 1
Please enter the Student 2 room: 3

Ahmad Esmaili and Samuel Stanley swapped.

Please select an option: P

Floor 1:
Room     Name          ID   Writeups
-----------------------------------------------------------
1     Ahmad Esmaili          214     0
2     Paul Fodor        114     0
3     Samuel Stanley    1     0

Please select an option: C

Please enter the source floor: 1
Please enter the destination floor: 3

Floor 1 Copied to Floor 3.

Please select an option: W

Please enter student name: Samuel Stanley

Samuel Stanley has 1 writeup.

Please select an option: P

Floor 1:
Room     Name          ID   Writeups
-----------------------------------------------------------
1     Ahmad Esmaili          214     0
2     Paul Fodor        114     0
3     Samuel Stanley    1     1

Please select an option: W

Please enter student name: Samuel Stanley

Samuel Stanley has 2 writeups.

Please select an option: W

Please enter student name: Samuel Stanley

Samuel Stanley has 3 writeups and has been removed from the building.
```

```
Please select an option: P

Floor 1:
Room       Name            ID   Writeups
----------------------------------------------------------
1       Ahmad Esmaili           214       0
2       Paul Fodor          114       0


Please select an option: F

Please select a floor: 3

Floor 3 selected.

Please select an option: P

Floor 1:
Room       Name            ID   Writeups
----------------------------------------------------------
1       Ahmad Esmaili           214       0
2       Paul Fodor          114       0
3       Samuel Stanley     1       0


Please select an option: R

Please select a student number: 2

Please select an option: F

Paul Fodor removed.

Please select an option: P

Floor 3:
Room       Name            ID   Writeups
----------------------------------------------------------
1       Ahmad Esmaili           214       0
2       Samuel Stanley     1       0

Please select an option: Q

See you later, alligator!
```

Exception Handling:
Your program should handle exceptions gracefully. It should not quit when an exception is thrown, inform the user, and allow the user to try making another input that will work (or return to the main menu).

Extra Credit

You will get up to 7 points extra credit for creating a JavaFX GUI, and up to 15 points for creating an Android App. Please discuss with your Grading TA if you choose to make an app or use Scene Builder, so you can coordinate submitting supplementary files with them.

Course Info | Schedule | Sections | Announcements | Homework | Exams | Help/FAQ | Grades | HOME