CSCI 221, Spring 2017 Due, Friday, February 24, 11:59pm
Programming HW 3 *Leniency* 24 hours later (see syllabus for details).

Develop a class that models a meeting room in a hotel or convention center.

Class Room **must** contain (private) variables that store (the data types must match what is shown):

```
String number; //a three digit room number, where first digit indicates the floor the room is located on
String name; // for those rooms with names, null otherwise
int capacity; // number of people the room can accommodate, expected to be > 0
char setUp;    // the way the meeting room is currently set up. Options are: 'T': theatre style,
               // 'C': classroom, 'U': U-shape, 'B' - banquet, 'E' – boardroom
boolean isReconfigurable; // true if the room setup can be changed, false otherwise
String configureOptions; // if room is reconfigurable options possible stored // as a String, ex
                         // "CU" for classroom and U-shape, else null
Room canBeCombinedWith; // initially set to be null, then changed to non-null later if this room can be
                        // combined with another room
static int numberOfRooms = 0; // count of all rooms in conference center
static int totalCapacity = 0; // capacity of all rooms combined
```

Class Room **must** contain the public constructor and methods described below (return type and signature must match what is shown below exactly):

A constructor that sets all instance variables and updates the static variables

```
public Room(String number, String name, int capacity, char setUp, boolean
   isReconfigurable, String configureOptions)
```

Initially, `canBeCombinedWith` should be set to null.

A method to update the canBeCombinedWith instance variable.

```
public void updateCanBeCombinedWith(Room r)
```

A method to return the capacity of a room.

```
public int getRoomCapacity( )
```

A method to update the configuration a room, if (1) the room is reconfigurable and (2) the option chosen is one of the configureOptions stored for this room. Otherwise, the room is not changed.

```
public void reconfigure(char option)
```

A method to determine if a room is compatible with another room (the parameter). Two rooms are *compatible* if they have the same capacity and, if reconfigurable, they have the same configure options.

```
public boolean isCompatibleWith(Room other)
```

A toString method to return a String representation of a room that when printed generates output shown below:

```
public String toString( )
```

```
Room number:    214
    capacity:   150
    setUp:      Classroom
    reconfigure options: Classroom, Banquet, U-shape.
    can be combined with room # 213
```

```
Room number:    103
    name:       Gold
    capacity:   100
    setUp:      Theatre
    can be combined with room # 104
```

where "reconfigure options" are shown only if a room is reconfigurable and "can be combined with" is shown only if the room can be combined with another room and name is only shown if a room has a name

A method to return the value stored in the numberOfRooms class variable.

```
public static int getNumberOfRooms
```

A method to return the value stored in the getTotalCapacity class variable.

```
public static int getTotalCapacity()
```

You can provide additional private helper methods, but they won't be tested directly.

PLEASE NOTE THAT I WILL BE OUT OF THE OFFICE ON THURSDAY, FEBRUARY 23!

A **`RoomClient`** class, to test your, class is provided. The output of the client should be:

```
Hotel has 0.
Hotel capacity is 0.

Room inventory:
Room number:      103
    name:   Gold
    capacity:     100
    setUp:  Theatre
    can be combined with room # 104

Room number:      104
    name:   Silver
    capacity:     50
    setUp:  Theatre
    can be combined with room # 103

Room number:      213
    capacity:     200
    setUp:  Classroom
    reconfigure options:  Classroom, Banquet, U-shape.

Room number:      214
    capacity:     100
    setUp:  Classroom
    reconfigure options:  Classroom, Banquet, U-shape.
    can be combined with room # 213

Room number:      215
    capacity:     100
    setUp:  Classroom
    reconfigure options:  Classroom, Banquet, U-shape.
    can be combined with room # 214

Room number:      350
    name:   Penthouse Boardroom
    capacity:     15
    setUp:  Boardroom

Hotel has 6 rooms.
Hotel capacity is 565 people.

Can room 215 be changed to theatre setup?
Before calling reconfigure method, room 215:
Room number:      215
    capacity:     100
    setUp:  Classroom
    reconfigure options:  Classroom, Banquet, U-shape.
    can be combined with room # 214

After reconfigure request, room 215 should show no change:
Room number:      215
    capacity:     100
    setUp:  Classroom
    reconfigure options:  Classroom, Banquet, U-shape.
    can be combined with room # 214


Can room 215 be changed to U-shape setup?
Before calling reconfigure method, room 215:
```

```
Room number:      215
    capacity:     100
    setUp:  Classroom
    reconfigure options:  Classroom, Banquet, U-shape.
    can be combined with room # 214

After reconfigure request, room 215 should show setup change:
Room number:      215
    capacity:     100
    setUp:  U-shape
    reconfigure options:  Classroom, Banquet, U-shape.
    can be combined with room # 214

Is room 215 compatible with room 214?
Should be true. It is: true

Is room 215 compatible with room 103?
Should be false. It is: false

---
```

Note – to create such a zip folder:

1. Create an empty folder with the name <lastname>HW3 (example McCauleyHW3)
2. Move your Room.java class into the folder and include the Room Client.java file as well.
3. Zip/compress it into <lastname>HW3.zip (example McCauleyHW3.zip)
4. Upload to OAKS.

To test if you have created a proper submission file, assuming your last name is Nero:

1. Unzip NeroHW3.zip
2. You should now see a folder named NeroHW3
3. Look in NeroHW2 and see the java and text files mentioned in 2 above.
4. If 1, 2, & 3 succeeded, success! Submit NeroHW3.zip

Grading Rubric

10 Points	Style: Comments and Indentation
40 Points	Functionality: • Program compiles (5) • Program runs (5) • Programs produce correct results (30)

- If the submitted program does not compile: 0 to 10 points
- If the submitted program compiles but does not run: 5 to 15 points
- If the submitted program compiles and runs: 10 to 40 points
- If the submitted program compiles, runs, and produces correct output: 40 of 40 points

The correctness of your program will be evaluated using test cases developed by the instructor.

Note: follow documentation/comments requirements shown in GeoLocation.java from HW2.

Late assignments will not be accepted – no exceptions (please do not email me your assignment after the due date, I will ignore it).

Please feel free to setup an appointment or drop by during office hours to discuss the assigned problem. I'll be more than happy to listen to your approach and make suggestions.

NOTE: I do not promise to be available to answer questions after 3:30pm on Fridays or over the weekend.

PLEASE NOTE THAT I WILL BE OUT OF THE OFFICE ON THURSDAY, FEBRUARY 23!