Programming and Data Structures
Homework Assignment #2 (OOP Applications)

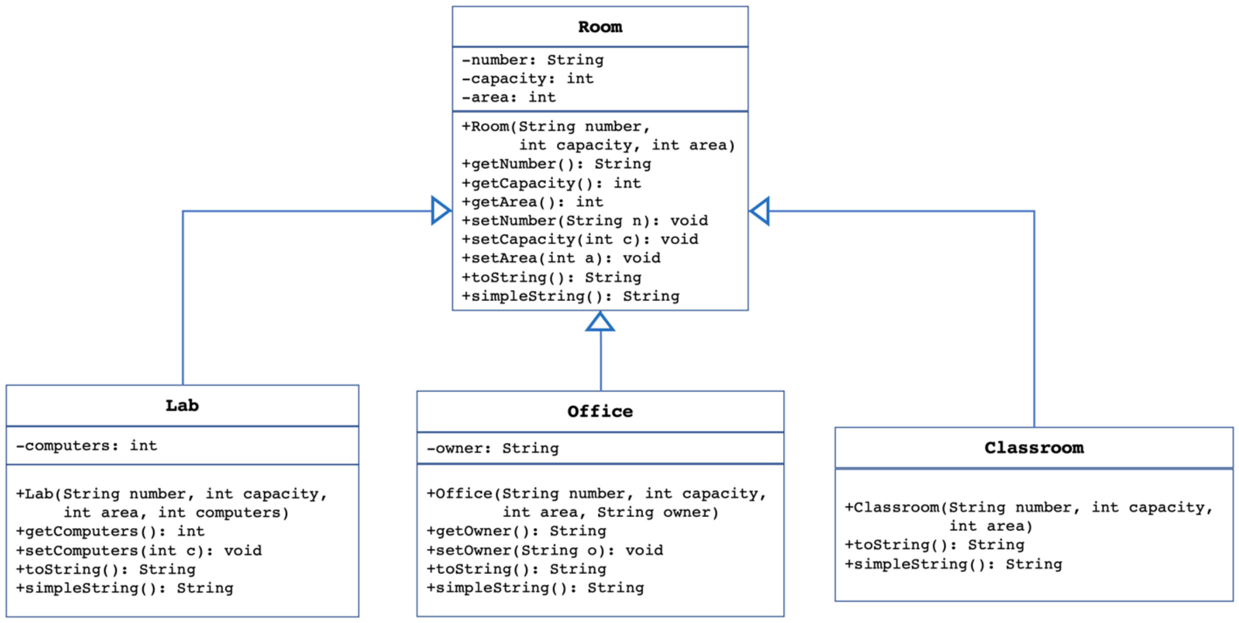
Assignment Objectives

At the end of this assignment, students should be able to:

1. Use regular expressions to validate user input
2. Create new Exception classes that extend Java class exceptions
3. Use Java Exception Handling mechanisms to throw and catch exceptions
4. Access text files for reading and writing

Assignment

1. Implement the class hierarchy shown in the UML diagram below. Use your code from homework 1 and add only the method ***simpleString()*** in each class. ***simpleString()*** should return a non-formatted string that contains the attributes of the room separated by a space.



2. Define a class ***InvalidRoomNumber*** that extends the class ***Exception***. The class must have two constructors, a default constructor and a constructor with one parameter of type ***String***.

3. Define a class ***Test*** to test the classes. Define the following static methods in the class ***Test***:

a. ***printRooms*** to print the first ***count*** elements of the array ***list***. The method has the following signature:

   public static void printRooms(Room[] list, int count)

b. ***findRoom*** to search for a room in the first ***count*** elements of the array ***list***. The method returns the index of the room if it is found, -1 otherwise. Here is the signature of the method:

   public static int findRoom(Room[] list, int count, String roomNumber)

c. ***sortRooms*** to sort the rooms in the first ***count*** elements of the array ***list*** based on the capacity. Here is the signature of the method:

   public static void sortRooms(Room[] list, int count)

   Use the insertion sort algorithm as provided below for type **int**. Modify the method to sort elements of type ***Room*** using the capacity to order the rooms.

```
public static void insertionSort(int[] list) {
 for (int i=1; i<list.length; i++) {
     //Insert element i in the sorted sub-list
     int currentVal = list[i];
     int j = i;
     while (j > 0 && currentVal < (list[j - 1])){
          // Shift element (j-1) into element (j)
          list[j] = list[j - 1];
          j--;
     }
     // Insert currentVal at position j
     list[j] = currentVal;
  }
}
```

d. ***checkRoomNumber*** to check the room number. The method accepts one parameter, the room number, and returns a boolean. A room number must have two alphabetical characters followed by three digits (PA101, BC200, CS001 are valid room numbers while Neville100, BC10, PAC101 are invalid). The method should return ***true*** if the room number is valid or throw an exception of type

InvalidRoomNumber otherwise. You must use regular expressions to check the room number.

e. Define the main method in class ***Test*** to do the following:
   i. Create an array ***rooms*** of type ***Room*** and size ***50***.
   ii. Initialize the array ***rooms*** using the list of rooms from the text file ***rooms.txt***. Remember to keep track of the number of rooms read from the file, which is the same as the number of elements added to the array ***rooms***. For example, ***rooms.txt*** contains 7 rooms. After reading the file, array ***rooms*** contains only 7 elements used (0 to 6). Elements 7 to 49 are all ***null***.
   iii. Prompt the user to select one of the following operations:
      1. ***View the list of rooms***: Display the list of rooms stored in the array ***rooms*** by calling the method ***printRooms***.
      2. ***Find a room:*** Prompt the user to enter a room number, check if it is valid, and search for it in the array ***rooms*** by calling the method ***findRoom***. Display an appropriate message to inform the user about the outcome of the search operation.
      3. ***Add a new room***: Prompt the user to enter a room number, check if it is valid, check if the room does not exist in the array ***rooms***, then prompt the user to enter the type of the room (lab, office, or classroom) and finally prompt the user to enter the attributes of the room. Use the attributes to create an instance of one of the classes (Lab, Office, or Classroom) and add the room to the array ***rooms***. Display an appropriate message to inform the user about the outcome of the operation.
      4. ***Remove an existing room***: Prompt the user to enter a room number, check if it is valid, find the room, and remove it from the

array ***rooms***. Display an appropriate message to inform the user about the outcome of the operation.

5. ***Sort the list of rooms***: call the method ***sortRooms*** to sort the elements used in the array ***rooms***. The rooms should be ordered from the lowest capacity to the highest.
6. ***Exit the program***: exit the program after saving the content of the array to the file "`rooms.txt`". Make sure you write to the file using the same format as the original file. ***Hint:*** use ***simpleString()***.

iv. In the main method, make sure you handle any exception of type ***InvalidRoomNumber*** that is thrown by ***checkRoomNumber***.

4. Test your program for all the operations. Two sample runs are provided below for testing.
5. Submit the files ***Room.java***, ***Lab.java***, ***Office.java***, ***Classroom.java***, ***InvalidRoomNumber.java***, and ***Test.java***. Javadoc comments are required.

=========================== **Sample run #1** ======================

```
Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity   Area       Owner/Computers
Office          BC208      1          25         Luke Vincent
Office          PA252      2          36         Houria Oudghiri
Classroom       PA101      20         45
Lab             BC101      38         67         25
Lab             PA100      47         120        40
Lab             PA110      65         150        45
Classroom       PA466      100        120

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
```

```
5: Sort rooms by capacity
6: Exit
2
Enter the room number:
BC100
Room not found.


Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
2
Enter the room number:
BC101
Room found: Lab            BC101     38          67          25

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
2
Enter the room number:
B100
Invalid room number. Must have 2 characters followed by 3 digits.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
3
Enter the room number:
PA466
Cannot add the room. Room found: Classroom  PA466     100   120

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
```

```
6: Exit
3
Enter the room number:
PA416
Enter the room type (lab/office/classroom):
classroom
Enter the room capacity (# of people):
120
Enter the room area (sq.ft):
150
Room PA416 successfully added.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity   Area       Owner/Computers
Office          BC208      1          25         Luke Vincent
Office          PA252      2          36         Houria Oudghiri
Classroom       PA101      20         45
Lab             BC101      38         67         25
Lab             PA100      47         120        40
Lab             PA110      65         150        45
Classroom       PA466      100        120
Classroom       PA416      120        150

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
3
Enter the room number:
NV200
Enter the room type (lab/office/classroom):
lab
Enter the room capacity (# of people):
60
Enter the room area (sq.ft):
80
Enter the # of computers:
45
Room NV200 successfully added.
```

```
Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity   Area       Owner/Computers
Office          BC208      1          25         Luke Vincent
Office          PA252      2          36         Houria Oudghiri
Classroom       PA101      20         45
Lab             BC101      38         67         25
Lab             PA100      47         120        40
Lab             PA110      65         150        45
Classroom       PA466      100        120
Classroom       PA416      120        150
Lab             NV200      60         80         45

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
3
Enter the room number:
PA308
Enter the room type (lab/office/classroom):
office
Enter the room capacity (# of people):
1
Enter the room area (sq.ft):
40
Enter the name of the owner (First and Last name):
Alice Boise
Room PA308 successfully added.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity   Area       Owner/Computers
Office          BC208      1          25         Luke Vincent
Office          PA252      2          36         Houria Oudghiri
```

```
Classroom       PA101      20         45
Lab             BC101      38         67         25
Lab             PA100      47         120        40
Lab             PA110      65         150        45
Classroom       PA466      100        120
Classroom       PA416      120        150
Lab             NV200      60         80         45
Office          PA308      1          40         Alice Boise

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
4
Enter the room number:
P101
Invalid room number. Must have 2 characters followed by 3 digits.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
4
Enter the room number:
PA111
Room PA111 not found.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
4
Enter the room number:
PA101
Room PA101 successfully removed.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
```

```
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity    Area        Owner/Computers
Office          BC208      1           25          Luke Vincent
Office          PA252      2           36          Houria Oudghiri
Lab             BC101      38          67          25
Lab             PA100      47          120         40
Lab             PA110      65          150         45
Classroom       PA466      100         120
Classroom       PA416      120         150
Lab             NV200      60          80          45
Office          PA308      1           40          Alice Boise

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
5

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity    Area        Owner/Computers
Office          BC208      1           25          Luke Vincent
Office          PA308      1           40          Alice Boise
Office          PA252      2           36          Houria Oudghiri
Lab             BC101      38          67          25
Lab             PA100      47          120         40
Lab             NV200      60          80          45
Lab             PA110      65          150         45
Classroom       PA466      100         120
Classroom       PA416      120         150

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
6
```

============================ **Sample run #2** ======================

```
/* Note how the modifications made in run #1 are
   saved in the file rooms.txt and read in run #2 */

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
Type            Number     Capacity   Area       Owner/Computers
Office          BC208      1          25         Luke Vincent
Office          PA308      1          40         Alice Boise
Office          PA252      2          36         Houria Oudghiri
Lab             BC101      38         67         25
Lab             PA100      47         120        40
Lab             NV200      60         80         45
Lab             PA110      65         150        45
Classroom       PA466      100        120
Classroom       PA416      120        150

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
3
Enter the room number:
PA208
Enter the room type (lab/office/classroom):
classroom
Enter the room capacity (# of people):
45
Enter the room area (sq ft):
80
Room PA208 successfully added.

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
1
```

```
Type            Number     Capacity   Area       Owner/Computers
Office          BC208      1          25         Luke Vincent
Office          PA308      1          40         Alice Boise
Office          PA252      2          36         Houria Oudghiri
Lab             BC101      38         67         25
Lab             PA100      47         120        40
Lab             NV200      60         80         45
Lab             PA110      65         150        45
Classroom       PA466      100        120
Classroom       PA416      120        150
Classroom       PA208      45         80

Select an operation:
1: View all rooms
2: Find a room
3: Add a new room
4: Remove an existing room
5: Sort rooms by capacity
6: Exit
6
```