```java
/** Functional :
 * Display study room information
 * reserve a study room
 * release a study room
 * concurrency handling
 * simulated student actions
 * error handling and exception
 * multiple test runs
 */
/** Non-Functional
 * concurrency management
 * execution requirments
 */


import java.util.*;

class StudyRoomUnavailableException extends Exception {
    public StudyRoomUnavailableException(String message) {
        super(message);
    }
}

class Studyroom {
    private int roomNumber;
    private int capacity;
    private boolean isAvailable;

    private final Object lock = new Object();

    Studyroom(int roomNumber, int capacity) {
        this.roomNumber = roomNumber;
        this.capacity = capacity;
        this.isAvailable = true;
    }
    //getting the roomnumber
    public int getRoom() {
        return roomNumber;
    }

    //getting the getcapacity
    public int getCapacity() {
        return capacity;
    }
```

```
    //getting the availability
    public boolean availability() {
        return isAvailable;
    }

    //checks reservation and reserve if available
    public void reservation() throws StudyRoomUnavailableException {
        synchronized (lock) {
        if (!isAvailable) {
            throw new StudyRoomUnavailableException("Study room " + roomNumber +
" is unavailable.");
        }
        // Now student is reserving the study room, so availiabilty is false
        isAvailable = false;


        // room is reserved
        System.out.println(Thread.currentThread().getName() + " reserved room " +
roomNumber);
        }
    }

    //release the room
    public void release() {
        synchronized (lock) {  // Ensure the safety of the thread
            isAvailable = true;
            System.out.println("Room " + roomNumber + " released.");
        }
    }
}

class Student extends Thread { //Lecture note
    private List<Studyroom> sR; // This sR contains the list of studyrooms
    private Random randomInt = new Random(); //Used Random class to generate a
random number between 0 and 3 as three studyrooms

    public Student(List<Studyroom> studyRooms, String name) {
        super(name); // supeer in thread
        this.sR = studyRooms;
    }

    @Override //overriding the run method in therad class
    public void run() {
        for (int i = 0; i < 3; i++) {   // for same student running three times
            try {
```

```
                Studyroom room = sR.get(randomInt.nextInt(sR.size()));
                room.reservation();
                Thread.sleep(2000);
                room.release();

            } catch (StudyRoomUnavailableException | InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " selected
room coudln't be reserved - " + e.getMessage());
            }
        }
    }
}



public class UniversityStudyRoomReservationSystemGroup_KeMora {
    public static void main(String[] args) {
        List<Studyroom> studyRooms = new ArrayList<>(); // This sR contains the
list of studyrooms
        studyRooms.add(new Studyroom(1, 4)); //Adding the rooms and capacity to
the list by ourelves
        studyRooms.add(new Studyroom(2, 6));
        studyRooms.add(new Studyroom(3, 8));

        Student s1 = new Student(studyRooms, "Student 1");
        Student s2 = new Student(studyRooms, "Student 2");
        Student s3 = new Student(studyRooms, "Student 3");

        s1.start();
        s2.start();
        s3.start();

    }
}
```

Output:

```
[Running] cd "c:\Users\Kalhara\Desktop\Java projects\inclass_3\" && javac KeMora_inclass_3.java && java KeMora_inclass_3
Student 3 reserved room 2
Student 1 reserved room 1
Student 2 selected room coudln't be reserved - Study room 2 is unavailable.
Student 2 reserved room 3
Room 1 released.
Room 2 released.
Student 3 reserved room 2
Room 3 released.
Student 2 reserved room 3
Student 1 selected room coudln't be reserved - Study room 3 is unavailable.
Student 1 reserved room 1
Room 2 released.
Student 3 selected room coudln't be reserved - Study room 1 is unavailable.
Room 3 released.
Room 1 released.

[Done] exited with code=0 in 4.792 seconds
```

```
[Running] cd "c:\Users\Kalhara\Desktop\Java projects\inclass_3\" && javac KeMora_inclass_3.java && java KeMora_inclass_3
Student 3 reserved room 1
Student 2 reserved room 2
Student 1 reserved room 3
Room 1 released.
Student 3 reserved room 1
Room 2 released.
Student 2 reserved room 2
Room 3 released.
Student 1 selected room coudln't be reserved - Study room 1 is unavailable.
Student 1 reserved room 3
Room 1 released.
Room 2 released.
Student 3 reserved room 2
Student 2 selected room coudln't be reserved - Study room 2 is unavailable.
Room 3 released.
Room 2 released.
```

```
[Running] cd "c:\Users\Kalhara\Desktop\Java projects\inclass_3\" && javac KeMora_inclass_3.java && java KeMora_inclass_3
Student 1 reserved room 2
Student 3 reserved room 3
Student 2 selected room coudln't be reserved - Study room 3 is unavailable.
Student 2 reserved room 1
Room 3 released.
Room 2 released.
Student 1 reserved room 2
Student 3 selected room coudln't be reserved - Study room 2 is unavailable.
Student 3 selected room coudln't be reserved - Study room 1 is unavailable.
Room 1 released.
Student 2 reserved room 3
Room 2 released.
Student 1 reserved room 2
Room 3 released.
Room 2 released.

[Done] exited with code=0 in 6.7 seconds
```

Observation:

For each runtime, it gets different outputs, because it is handle by the thread Scheduler.