Overview of the Problem

The ***Room Booking System is used by every collage in the university,*** it is designed for managing the operation of booking the rooms for events such as extra class, workshop, and meeting. All employees (staff and faculty) can book a room by the system, only staff are responsible for adding the booking requests to the system. So, if a faculty need to book a room, he/she must send request to any staff employee in order to make the booking.

Problem Description

1. The Application must read the information from the text file (named: INPUT.txt).
2. Make sure the file exits.
3. If the file doesn't exist, display the error message.

Error message must be:

```
run:
Input file INPUT.txt does not exist
```

4. The System will Read ALL data from the given input file and generate reports as given in output file sample.

The commands you will have to implement are as follows:

1. **ADD_ROOM: (NEED TO BE DONE)**

Makes a new room which is added to the system. The command will be followed by the following information:

The first number (3) after the command refers to an integer representing the number of rooms
[means system will accept ONLY THREE room record details]
The Next Line will have the following information:

- ✓ An integer representing room number
- ✓ An integer representing floor number which room located
- ✓ An integer representing capacity of the room
- ✓ A String representing room status (Available or Reserved)

As shown in the underneath example:

```
ADD_ROOM
3
106 1 40 Available
```

In the above command

Number of room = 3

Room Number = 106, Floor = 1, capacity = 40, Status = " Available"

In the output file that must be created by the application with name "**OUTPUT.txt**" must include the following message after each room data is inserted. See sample output file:

The Message: "***Command: Room 106 was added successfully***"

2. ADD_EVENT (DONE)

Makes a new event which is added to the system. The command will be followed by the following information:
The first number (3) after the command in the file refers to the number of events [means system will accept ONLY TREE event records details].
The Next Line will have the following information:

- ✓ A String representing event name
- ✓ A String representing event type (workshop, class, meeting)
- ✓ A date representing event date.
- ✓ A double representing starting time of event
- ✓ A double representing ending time of event
- ✓ An integer representing number of attendees

As shown in the underneath example:

```
ADD_EVENT
3
Cyber_Security workshop 969656400000 12:15 1:45 39
```

In the above command

Number of events = 3

Event name = Cyber_Security, Type = workshop, event date = 969656400000, starting time = 12:15, ending time = 1:45, Attendees number = 39

In the output file must include the following message after each event data is inserted. See sample output file:

The Message: "***Command: Event Cyber_Security was added successfully***"

3. ADD_STAFF: (DONE)

Makes a new staff which is added to the system. The command will be followed by the following information:

The first number (2) after the command in the file refers to the number of staff
[means system will accept ONLY TWO staff records details].
The Next Line will have the following information:

- ✓ An integer representing employee id
- ✓ A String representing First Name
- ✓ A String representing Last Name
- ✓ An integer representing birth year
- ✓ An integer representing employment year
- ✓ A String representing department
- ✓ A String representing position

As shown in the underneath example:

```
ADD_STAFF
2
12132 Abeer Alsalem 1989 2014 IT Programmer
```

In the above command

Number of staff = 2

Employee id = 12132, First Name= Abeer, Last Name= Alsalem, Birth year = 1989, Employment year =2014, department = IT, position = programmer

In the output file must include the following message after each staff data is inserted. See sample output file:

The Message: "***Command: Staff Employee with ID 12132 was added successfully***"

4. ADD_FACILTY: (DONE)

Makes a new faculty which is added to the system. The command will be followed by the following information:

The first number (1) after the command in the file refers to the number of faculty
[means system will accept ONLY ONE faculty records details].
The Next Line will have the following information:

- ✓ An integer representing employee id
- ✓ A String representing First Name
- ✓ A String representing Last Name
- ✓ An integer representing birth year
- ✓ An integer representing employment year
- ✓ A String representing department
- ✓ A String representing degree

As shown in the underneath example:

```
ADD_FACILTY
1
21861 Asma Kamel 1979 2008 CS Lecturer
```

In the above command

Number of faculty = 1

Employee id = 21861, First Name= Asma, Last Name= Kamel, Birth year = 1979, Employment year =2008, department = CS, position = Lecturer

In the output file must include the following message after each staff data is inserted. See sample output file:

The Message: "***Command: Faculty Employee with ID 21861 was added successfully***"

5. BOOKING: (NEED TO BE DONE)

Makes a new booking which is added to the system. The command will be followed by the following information:
The first number (3) after the command refers to the number of booking [means system will accept ONLY THREE booking records details]. The Next will have the following information:

- ✓ An integer representing room number
- ✓ A string representing event name
- ✓ An integer representing employee id

As shown in the underneath example:

```
BOOKING
3
111 Cyber_Security 12132
```

In the above command

Number of booking = 3

Room number = 111, event name= Cyber_Security, employee id= 12132

In the output file must include the following message after each Aircraft data is inserted. See sample output file:

The Message: "***Booking with Number (274) is created and added to booking array list***)"

Important Notes:

- You need to check the capacity of the requested room if it can NOT accommodate the number of attendees of the event and show appropriate message as in the underneath examples.
- You need to check if the employee who want to book a room is a STAFF EMPLOYEE, if employee is faculty employee, show appropriate message as in the underneath examples.
- Make sure to make use of the THREE provided methods: (searchRoomIndex – searchEventIndex - searchEmployeeIndex).
- These methods will search in the ArrayLists of room, ArrayList of event, and ArrayList of employee and return its index.
- If capacity of room accommodate the number of attendees of the event and the moderator (employee who request to book the room) is staff employee, you can now make the booking by generate booking number, create the booking object, and add it to Booking ArrayList.
- Make sure to change the status of room to (Reserved) after booking it.
- Print booking information of all booking objects in booking ArrayList.

Example 1:	Example 2:	Example 3:
Command in input file: 111 Cyber_Security 12132 **Feedback message in output file:** `Unsuccessful Booking Attendees number is exceed the Room 111 capacity`	**Command in input file:** 106 Data_Science 21861 **Feedback message in output file:** `Unsuccessful Booking Employee with ID 21861 is not a Staff Employee`	**Command in input file:** 212 Collage_councel 10561 **Feedback message in output file:** `Booking with Number (274) is created and added to booking array list`

Further Details are as follows:

1. The Application must define 4 ArrayList: ArrayList to save all Room objects and name it '**room**', ArrayList to save all Event objects and name it '**event**', ArrayList to save all Employee objects and name it '**employee**', and the last ArrayList to save all Booking objects and name it '**booking**'
2. The application must define seven classes:

   (**Room / Event / Employee / Staff / Faculty / Booking / FINAL-P1**(main class))
   Follow the relations and attributes given in the UML diagram below.

3. You need to apply all the relationships shown in the UML correctly especially in the following classes: Staff and Faculty.

4. Your program output must be exactly same as given sample output files.

5. **See the UML Diagram to know basic class details (Attached also with exam folder)**