The purpose of the program is to develop a reservation system for a transit system.
The client request list of available seats from server, and then reserves the seats by sending the seat number to the server.

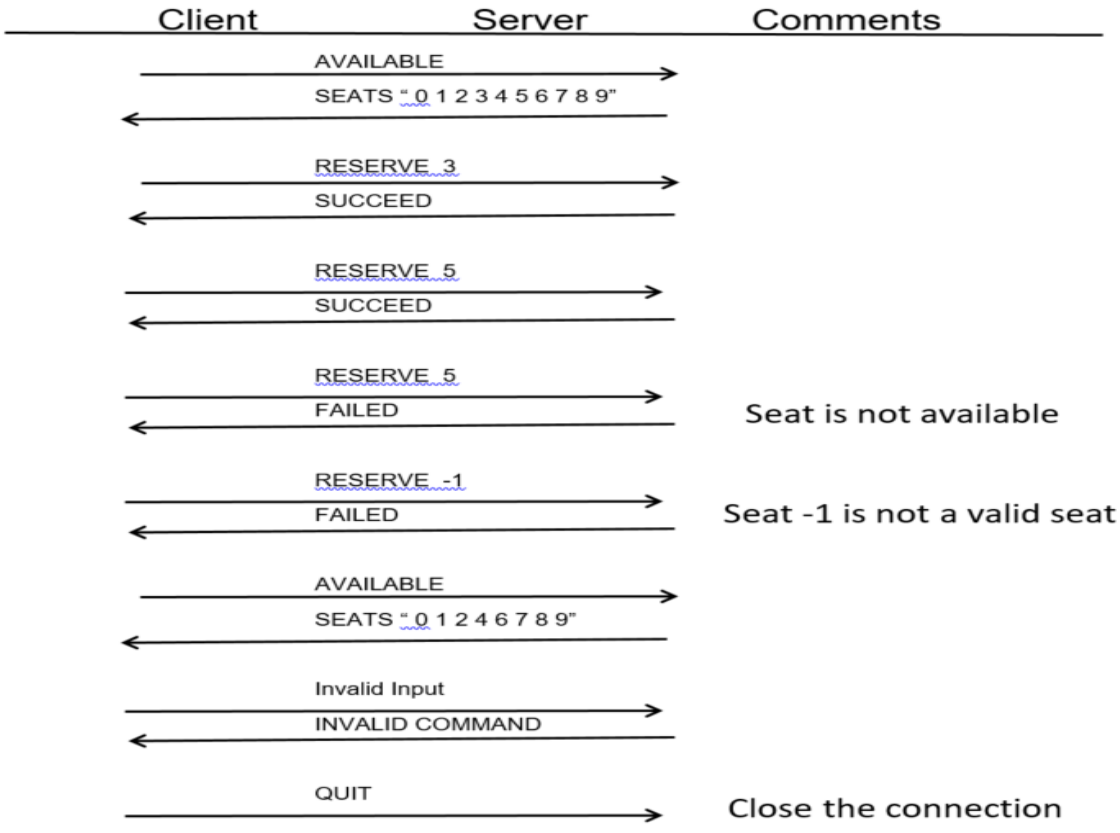
For simplicity, we assume that

1. Client can reserve as many seats as required, but it can reserve them one by one.
2. The seats are listed as an array of boolean variables. True values of the array stand for reserved seats and false values for available seats.
3. Server does not track the reservation process. (It does not know who reserved what).

Here is a description of the protocol that client and server use.

Client Request	Server Response	Description
AVAILABLE	***SEATS*** *listOfSeats(String)*	Server response with list of available seats separated by space as a String.
RESERVE seatNumber(int)	**SUCCEED** **FAILED**	Server reserves the seat number and responses as SUCCEED. If the seat is already reserved, or the requested seat number is not valid then server responses as FAILED.
QUIT		The connection closes
anything else	**INVALID COMMAND**	Server waits for another request

A Sample session of Client/Server connection is shown below for your reference only



For this question, you are given three complete classes:

1. The complete code for **class Bus**. Please do not modify this class.
   The class handles its own reservation process by using an array of boolean variables as seats. True value means the seat is already reserved, and false means that the seat is available. Each seat is identified by its index number. Bus has two methods:
   - `public String getAvailableSeats()` returns index of the available seats separated by a space as a string. If the bus has no available seats, then it returns string "Full" that indicates there is no seat left.
   - `boolean reserve(int index)` reserves the seat at given index of the array by setting it to true, and then returns true. If the seat is already reserved, or if the index is not a valid index it returns false.

2. The complete code for **BusClient** that implements the correct protocol and is fully developed. The BusClient class provides a User Interface for users to send a request to the Bus server. Do not modify this class.

3. **class BusServer** is fully developed and do not modify this class.

You are asked to implement the class **BusService**.

Here are the tasks to do (not necessarily in the provided order)

- create a new appropriate class BusService
- the class opens the **DataInputStream**, **DataOutputStream** from the open socket**.**
- while client has not sent QUIT request
  - read the command from the client based on the protocol
  - If command is AVAILABLE, then call getAvailableSeats() of the bus and send the result to the client based on the protocol
  - If command is RESERVE seatNumber, then call reserve(seatNumner) of the *bus* (an instance of class Bus) and send the result to the client based on the protocol
  - If the command is none of the protocols (AVAILABLE, RESERVE, or QUIT), send "INVALID COMMAND" to the client.
  - If command is QUIT, then terminate the connection.

HINT:

- Use writeUTF and readUTF methods to send/receive strings.
- Use writeInt and readInt methods to send/receive integer numbers.