Assignment 2. Java Programming Language, CSE3040 & AIE3052

Student Name:
Student ID:

Goal

- Create “TicketServer.java” and “TicketClient.java” using the contents we learned in class to allow multiple clients to reserve a seat by purchasing simultaneously.

Task Requirements:

- Each seat is assigned a unique number.
- If the client requests to reserve a seat, the server is required to send the client the data of the remaining seats.
- If another client takes the seat while making a reservation, the server should allow the client to choose which seat to reserve again.
- Synchronization should work as follows:
    a. client requests for a reservation
    b. server checks if that seat is available
    c. if yes, remove it from the list and print "Seat seat_number* booked successfully! Remaining seats: available_seats*”
    d. if no, print "Seat seat_number* is already taken. Please choose another seat."
- Available seats should be updated after the reservation is complete.
- The client will spend some time before they decide where to sit.

Submission Guide Line:

- Submit two Java files named “TicketServer.java” and “TicketClient.java”
- Both files should be compiled into zip or tar.gz with the file name “java_assignment2_studentID_name” where studentID section should be your individual student ID.
- Incorrect submission format will lead to -25% penalty.

How the Assignment will be Graded:

- Both Java files will be executed in a local environment.
- TicketServer.java will be executed first then TicketClient will be executed afterwards.

Base Code for TicketServer.java:

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.concurrent.*;

public class TicketServer {
    // List to maintain available seats
    private static final List<Integer> availableSeats = new
ArrayList<>();
    // Lock object for synchronization
    private static final Object lock = new Object();

    static {
        // TODO: initialize 20 unique seats
        }
    }

    public static void main(String[] args) {
        // TODO: create new socket with port 8080 and limit
thread count to 5

            while (true) {
                // Client connected. should be printed if
client access the server.
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    static class TicketHandler implements Runnable {
        private final Socket clientSocket;

        public TicketHandler(Socket clientSocket) {
            this.clientSocket = clientSocket;
        }

        @Override
        public void run() {
            try (BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                 PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true)) {

                boolean seatBooked = false;

                while (!seatBooked) {
                    // TODO:
                    // Step 1: Show available seats to the
client

                    // Step 2: Receive the chosen seat
number from the client

                    // Step 3: Attempt to book the chosen
seat
                    synchronized (lock) {
                        // TODO: Check if the seat is
available and book it if possible
                        // Send success or failure messages
back to the client
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                // TODO: Client disconnected. should be
printed if the client is off the server.
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Base code for TicketClient.java

```
import java.io.*;
import java.net.*;
import java.util.Random;

public class TicketClient {
    public static void main(String[] args) {
    // TODO: create 5 clients

    }

    static class ClientTask implements Runnable {
        private final String clientName;

        public ClientTask(String clientName) {
            this.clientName = clientName;
        }

        @Override
        public void run() {
        // TODO: make a connection with the server as localhost and print
“client_name* connected to the server.”
                boolean seatBooked = false;

                while (!seatBooked) {
                    // TODO:
                    // Step 1: Receive and display available seats from the
server

                    // Step 2: Choose a random seat from the available seats

                    // Step 3: Send the chosen seat to the server

                    // Step 4: Receive booking confirmation or error
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    private void sleepRandomTime() {
        try {
        // TODO: the client will wait for 5 to 10 seconds (use Random())
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```