Distributed Systems
(SOFE 4790U)

Assignment 2

Professor: Dr. Anwar Abdalbari

Name	ID	Initial
Syed Airaj Hussain	100789134	A.H

Due Date: November 23, 2024

1. Application idea

The program is a game where the user must guess a number from 1-100 randomly generated each time a new game is played. The user chooses the option to start the game, and with each attempt, the server replies whether the number guess is higher or lower compared to the random number. Once the user finally gets the number, the game is ended and can create a new game. A leaderboard is created and updated based on the user who got the number with the least guesses. Even though the highest person on the leaderboard requires getting the guess within the first try, the leaderboard is updated with the latest winner, even if the previous guesser got it on the first try.

2. Describe the core functionalities (5 unique functions)

1 Starting a new game:
The game is initialized and the server generates a ‘secret’ random number as well as resetting any attempts that were held by the previous run games. This function ensures that the game is started if there are no current running games. The number generated is from 1-100, the attempts are reset on a newly started game, and the user is prevented from starting a new game if one is currently in progress.

2 Making a guess:
The user can make a guess, the guess is compared with the randomly generated number. Depending on the number guess the user is prompted with a too high/too low message or correct, giving them insight on how to make their next guess. The leaderboard is updated if they guess correctly with their number of attempts.

3 Leaderboard Management:
The user can view the current leaderboard status which displays the number of tries to get the correct answer, the leaderboard holds the correct guessers with their attempts and is updated with the latest winner. The winner with the least amount of attempts is displayed at the top. The least number of attempts possible is 1, the leaderboard is updated with the most current winner, hence if a player previously got the correct answer within 1 attempt they are replaced with the most recent winner if they got the correct number in 1 attempt.

4 Get Game Status:
The user is able to see their current progress, the game status displays whether there is a game running or not, and it displays the user's current attempts of that game.

5 End Game:
The user can end the current game they are in. This function stops the current game and does not update the leaderboard, this function is for when the user isn’t satisfied with the amount of attempts they’ve taken to guess the password and can end the game and retry guessing the number. An output is sent to the user displaying the game has ended, their current attempts are reset and a new number is generated.

3. Challenges and solutions

1 Dealing with player data in the leaderboard:

It was difficult to find a way to create a leaderboard that sorts itself based on the number of attempts the user has made. A solution to this was having an array list that holds their names and values, with a for loop that would loop through the list and sort it based on the least amount of attempts as well as the most recent user.

2 Tracking User Attempts:

Making sure the users attempts were correctly tracked was difficult, reason is because the user could restart or end the game, so a method was created to combat this which was the that the attempts would be tracked so long as the game was ‘live’, however if the user pressed the option to end the game, it would clear their attempts and not store them.

3 Coordinating Players Actions with Game State:

Ensuring that the players actions are in sync with the game's current state was important as it could affect their attempts and the leaderboard. This was fixed by added a ‘gameInProgess’ method that would flag whether the user is in a game or not, if the user was still in a game their progress was collected, however if it were to be ended with the exit option, the leaderboard wouldn’t be affected and their attempts were reset.

4 Managing RMI Calls

A main challenge was to make sure that there is correct communication between the server and client all while using RMI, especially when transmitting data like usernames, the users guesses and updates to the leaderboard.

A solution to this was having an interface ‘IGuessTheNumber’ that has several methods to handle RMI inputs, such as setting the users name and accepting their name/guess.

4. Testing

Initial Setup of Program:

Fig 4.1.1- Running rmi registry at [port]

Fig 4.1.2- Server Running

Fig 4.1.3- Client Running

First User Running:

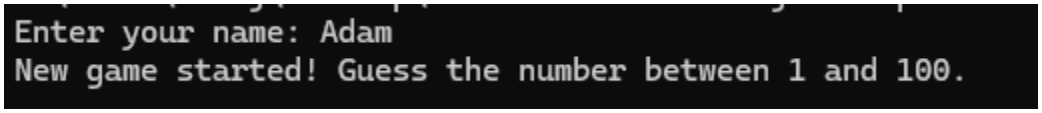

Fig 4.2.1- User enters name

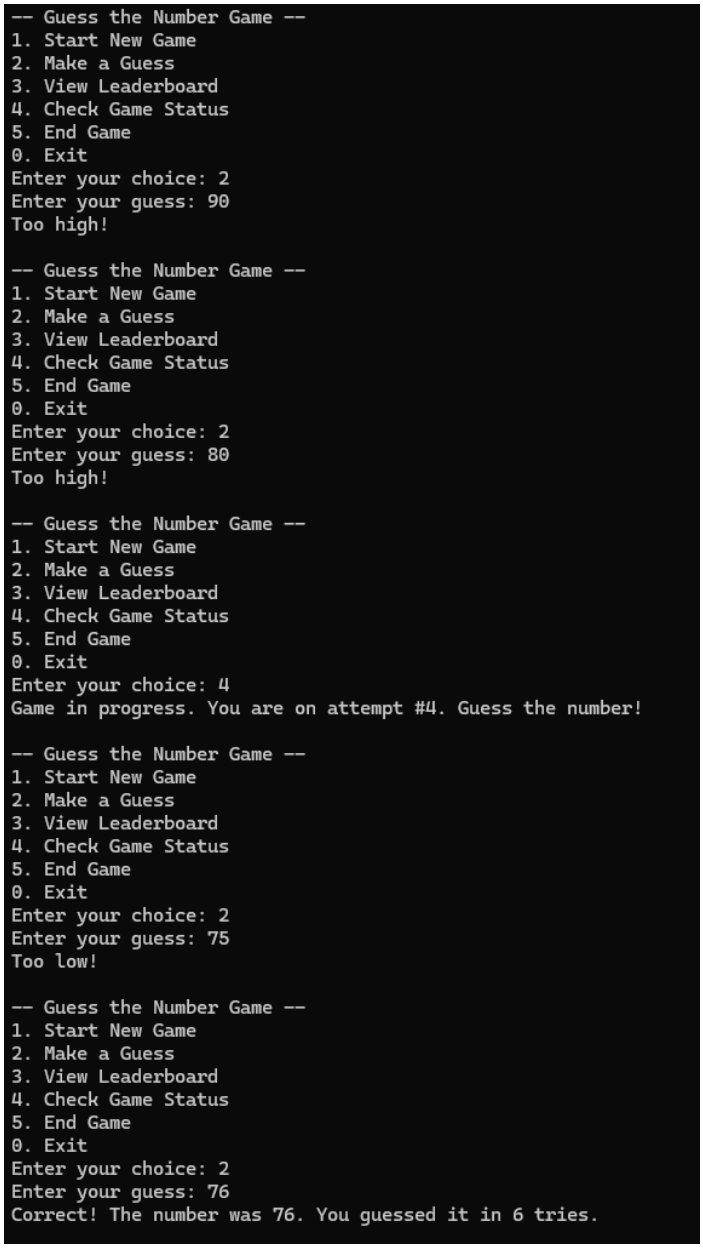

Fig 4.2.2- User guessing

The user consistently keeps guessing, and during they guessing they choose 4- Check Game Status, which displays from the serve that they're on attempt #4, as well as they're consistent attempts to display whether they are too low or too high from the secret number.

```
-- Guess the Number Game --
1. Start New Game
2. Make a Guess
3. View Leaderboard
4. Check Game Status
5. End Game
0. Exit
Enter your choice: 2
Enter your guess: 76
Correct! The number was 76. You guessed it in 6 tries.

-- Guess the Number Game --
1. Start New Game
2. Make a Guess
3. View Leaderboard
4. Check Game Status
5. End Game
0. Exit
Enter your choice: 3
Leaderboard:
Adam - 6 attempts
Jake - 6 attempts
Moe - 7 attempts
Sam - 9 attempts
apple - 11 attempts
```

Fig 4.2.3- User Leaderboard

Despite the last user Jake having 6 attempts, the current user, Adam, has had 6 attempts as well however he is placed at the top since he is the most recent winner. The leaderboard displays all the previous attempted tries by other clients that are accessing the server at the same time.

Link to GitHub repository:

https://github.com/AirajHussain/GuessTheNumber

Link to google drive of video tutorial:

https://drive.google.com/file/d/1qA9BTETWwD8nk1c4QLC1gD7G6kj4N70C/view?usp=sharing