[CS209A-24Spring] Assignment 2 (100 Points)

Question Design: Yao Zhao
Code Sample: Qiujiang Chen
Evaluation: Haihan Zhang

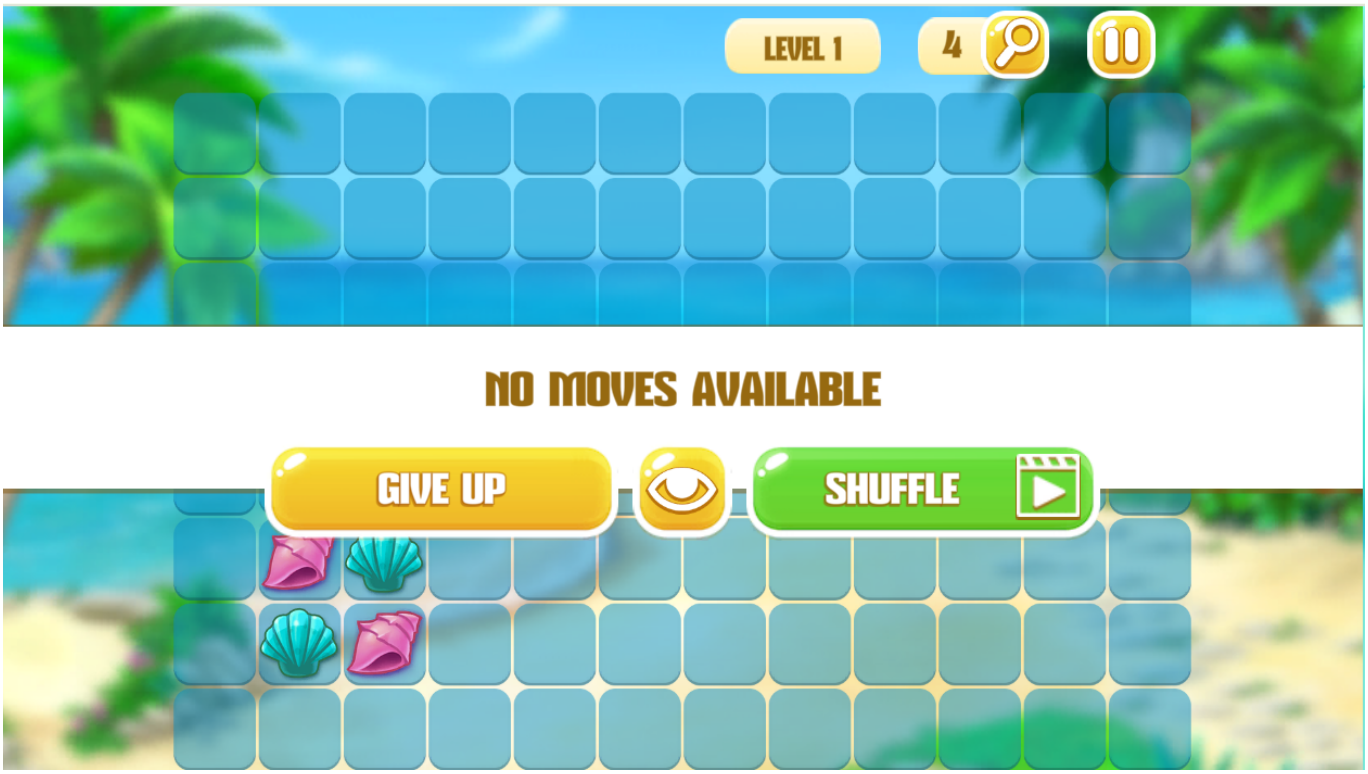
Deadline: 11:55 PM, Nov. 18

Late submissions will NOT be accepted after the deadline.

Overview: Linking Game

In this assignment, you will implement a **two-player version** of the **"Linking Game"** (also known as **"Lianliankan"**) using **JavaFX** for the graphical interface and **Java Socket Programming** for network communication. The two-player version of the "Linking Game" is a competitive puzzle game where two players are connected via a **client-server architecture** and attempt to match pairs of identical icons that can be connected **with three or less straight lines** on a grid.

Here are some game diagrams:



In the two-player mode, players **take turns** selecting matching icons. Each successful match earns points. The player with the higher score at the end wins.

The detailed requirements are outlined below.

Key Requirements

The implementation will involve:

- **Socket Programming**
- **Multithreading**
- **JavaFX** for the graphical interface

1. Server-Side Implementation (35 Points)

The server will manage game states, handle client connections, and ensure proper synchronization between clients and server.

Responsibilities:

- **Player Connection Management:**
  - The server should wait for new players and automatically matches a newly connected player to another player to start a game. If no other player is available, the server should inform the newly connected player to wait.
  - Once a game starts, the server should generate a game board with some randomly distributed pairs of icons (represented by numbers) based on board size and sends it to both players simultaneously.
- **Game Management:** end when no more matching move
  - Track the status of each game session and notify players when a game is won, lost, or tied.

Communication Protocol:

You are required to design a robust communication protocol between the server and clients to ensure smooth and synchronized gameplay.

---

2. Client-Side Implementation (35 Points)

The client program will **handle game interactions** and **facilitate communication with the server**.

Responsibilities:

- **Connection:**
  - Players connect to the server and wait to be matched with another player.
  - Once matched, the two players alternate turns attempting to link two matching icons on the board.
- **Gameplay:**
  - Players select two icons that they believe can be linked with no more than two right-angled turns.
  - Check if the connection is valid.
  - Ensure that the GUI correctly reflects whether the connection is valid.

- **Synchronization with the Server:**
  - The state of the game board and the player's score should be promptly synchronized with the server to ensure accurate tracking of the game state.
- **Notifications:**
  - Players should receive notifications from the server when they win, lose, or tie the game.

---

3. Graphical User Interface (GUI) Implementation (15 Points)

The game's GUI will be developed using **JavaFX** and should meet the following requirements:

Components:

- **Game Board:**
  - The grid should display randomly distributed pairs of icons (e.g., fruits, animals) as provided by the server.
  - Grid sizes can vary (4x4, 6x8, 8x8, etc.) and should be customizable by the players.
  - Players can select two identical icons. If the client determines that it is a valid connection, those lines should be drawn, and both the icons and lines should then disappear.
- **Score:**
  - Display the player's current score.
- **Game Result:**
  - Display the player's win, loss, or tie status when the game ends.

Note:

- The GUI must be implemented using **JavaFX**. No points will be awarded for using console-based outputs or other GUI frameworks (e.g., Swing, Vue).

---

4. Exception Handling (15 Points)

The game should gracefully handle potential errors, including:

- Server crashes
- Player disconnections (whether intentional or accidental)

Your program should account for these exceptions and handle them appropriately to minimize impact on user experience.

---

5. Bonus Features (15 Points)

- **Account Management & Game History (5 Points):**
  - Implement user registration and login.

    - Allow players to view their own and others' game history and online status.
- **Opponent selection (5 Points):**
    - After connecting to the server, a player is able to see the list of players that are currently waiting to be matched. The player could choose the opponent from the list of players to start a new game.
- **Game Resumption (5 points):**
    - If a player quits a game intentionally or accidentally, they should be able to reconnect to the server and resume the game from where they left off, with the same opponent if available.

Additional Clarifications

1. **Player Matching Rule:** Does the server always match a player with the first waiting player?
    - No specific requirement exists to match a player with the first waiting player. If there are more than two players in the queue, the first two players will be matched. In high-concurrency scenarios, if the server is overloaded, players may be added to the queue and not necessarily matched with the first waiting player. This situation can be challenging to test.
2. **Score Calculation of the players:** Does the score depend on the number of lines used to connect the icons?
    - The score is not affected by the number of lines used to connect the icons; as long as the connection is valid, the score remains the same. However, if students design a scoring system based on the number of lines, this is also acceptable.
3. **Game End Condition:** Does the game end when there are no connectable icons left?
    - The game can be set to end when there are no connectable icons left, or the server can automatically shuffle the board to allow continued play. Additionally, users can choose to end the game or shuffle again. The choice of approach has no effect on the grading.

Demonstration

A sample code for the single-player version of the game is provided to help you get started.
Link to Code Sample

Suggestion

- Start by completing the single-player version:
    - Implement the option to customize the **board size** before the game begins.
    - The grid should display randomly distributed pairs of icons based on the selected **board size**.
    - Players can select two identical icons. If the connection is valid (as validated by the check code in the provided sample), draw those lines, and then let the icons and lines disappear.
    - Update the player's score throughout the game.
    - Display the result when the game ends.
- During testing, you will earn 15 points (equivalent to the points for requirement 3) for demonstrating that the single-player version functions correctly. Note that the scoring for the GUI in the two-player

mode will not be counted again.

Submission Guidelines

- Submit all project files as a single ZIP file named `A2-yourStudentID.zip`.
- Upload the ZIP file to Blackboard by the deadline.

Evaluation Process

- **Functionality Testing:** You will demonstrate your project during the lab session on **Nov. 19-20 (Week 11)**.
- We will verify that the required functionalities have been correctly implemented.
