COMP3211- Project: command-line-based Jungle game

Group 104

Cheung Kei Yau, Chong Zhi Yi, Lau Ming Yin, Lee Yin Lam Elaine

Supported user commands

Starting Screen:

```
Enter command: start | load | replay | exit
```

Start: starts a new game.
Load: loads a saved game (.jungle) to continue playing.
Replay: replays the past record of a game (.record).
Exit: exits the program.

Replay mode:
Auto: automatically replayed
Step: replay record step by step manually
Cancel: exit replay mode

```
Enter record number (e.g. 1, 11), or leave empty to go back to previous page: 11
Attempting to load: record\game_log_11.record
Loaded 17 commands.
Replay mode? (auto | step | cancel):
```

Once start game /load game → game loop

```
Enter command (move [srcPos] [desPos] | undo | save | stop)
```

Move: move a piece from [source position] to [destination position]
Undo: revert the previous move and reduce undo count by 1
Save: save the game to the folder jungle
Stop: exit game and return starting screen

Output of the game status

An ASCII code of 7x9 game board is printed on screen with animal pieces on the starting squares, traps, rivers and dens.

Error handling mechanism

- Purpose：prevent malform commands in the .record file

```
    try {
        chess.GameSetUp(rev: true);
    } catch (RuntimeException e) {
        System.err.println("Failed to start game during replay: " + e.getMessage());
    }
    gameStarted = true;
} else if (lower.startsWith(prefix: "move ")) {
    // ensure the game is initialized before applying moves
    if (!gameStarted) {
        System.out.println(x: "No explicit start found in record - starting game with random names.
        chess.setRandomPlayerNames();
        try {
            chess.GameSetUp(rev: true);
        } catch (RuntimeException e) {
            System.err.println("Failed to initialize game for replay: " + e.getMessage());
```

```
try {
    chess.showMap();
} catch (RuntimeException ignore) { }

if (chess.isGameOver()) {
    System.out.println("Game finished during replay. Winner: " + chess.getWinnerName());
    break;
}
if (stopReplay) break;

if (mode.equals(anObject: "step")) {
    System.out.print(s: "Press Enter to continue (or 'cancel' to stop replay): ");
    try {
        String stepInput = scanner.nextLine().trim().toLowerCase();
        if (stepInput.equals(anObject: "cancel")) {
            System.out.println(x: "Replay cancelled by user.");
            return;
        }
    } catch (NoSuchElementException | IllegalStateException e) {
        System.err.println("Input error while stepping: " + e.getMessage());
        return;
```

```
                try {
                    chess.movePiece(fromRow, fromCol, toRow, toCol);
                } catch (RuntimeException e) {
                    System.err.println("Failed to apply move during replay: " + e.getMessage());
                }
            } catch (IndexOutOfBoundsException | IllegalArgumentException e) {
                System.out.println("Skipping malformed move: " + cmdLine);
            }
        } else {
            System.out.println("Skipping malformed move: " + cmdLine);
        }
    } else if (lower.equals(anObject: "undo")) {
        if (!gameStarted) {
            System.out.println(x: "Ignoring undo before game start.");
        } else {
            try {
                chess.undo();
            } catch (RuntimeException e) {
                System.err.println("Failed to undo during replay: " + e.getMessage());
            }
        }
    } else if (lower.equals(anObject: "stop")) {
        System.out.println(x: "Stop command encountered in record.");
        stopReplay = true;
    } else {
        System.out.println("Unknown command in record, ignoring: " + cmdLine);
    }
} catch (RuntimeException e) {
    System.out.println("Error executing command '" + cmdLine + "': " + e.getMessage());
```

prevent malform commands

- To prevent someone edit the files and cause parsing errors

record > game_log_3.record

```
1   start Simon Thaddeus
2   move E7 D7
3   move E3 E2
4   move D7 D6
5   move G3 G4
6   move D6 D5
7   move G1 G2
8   move D5 D4
9   move G4 G5
10  move D4 D3
11  move E2 E3
12  move D3 D2
13  move E3 F3
14  move C7 D7
15  move F2 E2
16  move D2 D1
17  Simon (Red) wins by entering Blue's den!
```

Original file

record > game_log_3.record

```
1   Simon Thaddeus
2   move E7 H7
3   move E3 E0
4   move D7 D6
5   move A7 A6 A5
6   move G3 G4
7   move D6 D5
8   move G1 G2
9   move D5 D4
10  move G4 G5
11  move D4 D3
12  move E2 E3
13  move D3 D2
14  move E3 F3
15  move C7 D7
16  move F2 E2
17  move D2 H1
18  May I stop the game?
19  stop
20  Simon (Red) wins by entering Blue's den!
```

Edited file with malform commands

Output for the replay mode with this file:

```
>> Simon Thaddeus
Unknown command in record, ignoring: Simon Thaddeus
```

```
>> move E7 H7
No explicit start found in record - starting game with random names.
E7 -> H7
Failed to apply move during replay: Index 7 out of bounds for length 7
```

```
>> move E3 E0
E3 -> E0
Cannot move into action row.
```

```
>> move D7 D6
D7 -> D6
No piece at the source position.
```

```
>> move A7 A6 A5
Skipping malformed move: move A7 A6 A5
```

```
>> May I stop the game?
Unknown command in record, ignoring: May I stop the game?
```

```
>> stop
Stop command encountered in record.
```

record > game_log_3.record

```
 1   Simon Thaddeus
 2   move E7 H7
 3   move E3 E0
 4   move D7 D6
 5   move A7 A6 A5
 6   move G3 G4
 7   move D6 D5
 8   move G1 G2
 9   move D5 D4
10   move G4 G5
11   move D4 D3
12   move E2 E3
13   move D3 D2
14   move E3 F3
15   move C7 D7
16   move F2 E2
17   move D2 H1
18   May I stop the game?
19   stop
20   Simon (Red) wins by entering Blue's den!
```

Edited file with malform commands

prevent malform commands

- MAX_UNDO = 3;
- To prevent anyone editing the files and cause parsing errors

jungle > game_save_3.jungle

```
1   player RED I'm red
2   player BLUE Bartholomew
3   current BLUE
4   undos RED -1
5   undos BLUE 4
```

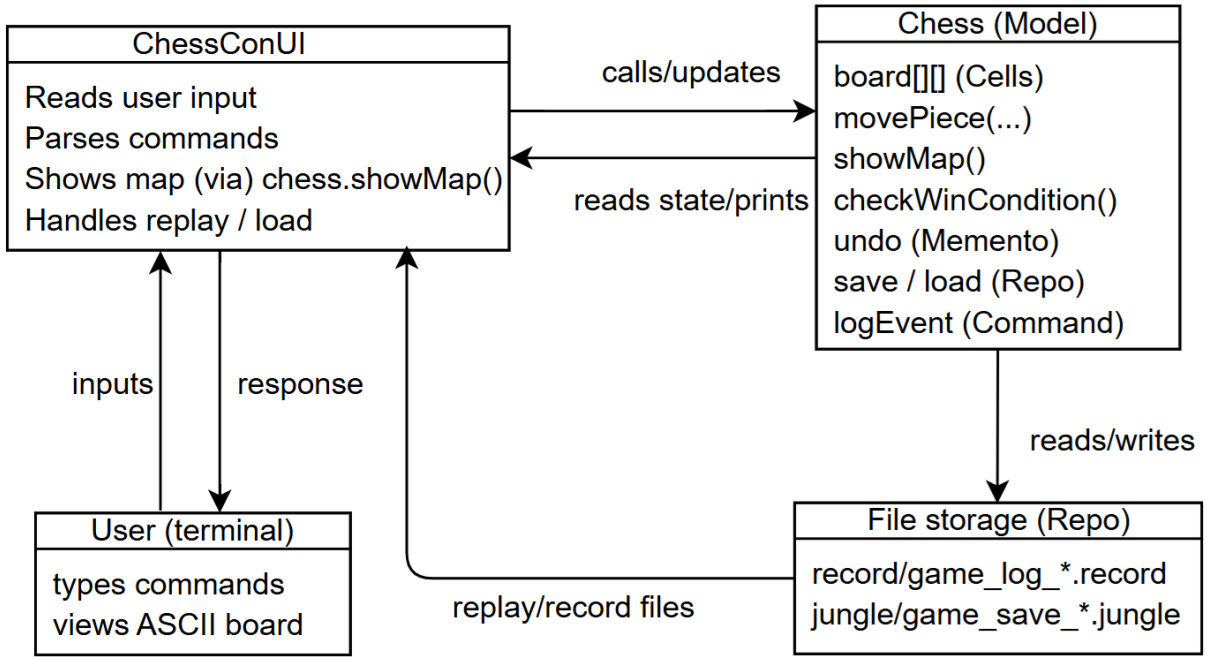
```
Remaining pieces: I'm red (Red): 8, Bartholomew (Blue): 8
Remaining undo: I'm red (Red): 0, Bartholomew (Blue): 3
Current player: Bartholomew (BLUE)
Next player: I'm red (RED)
```

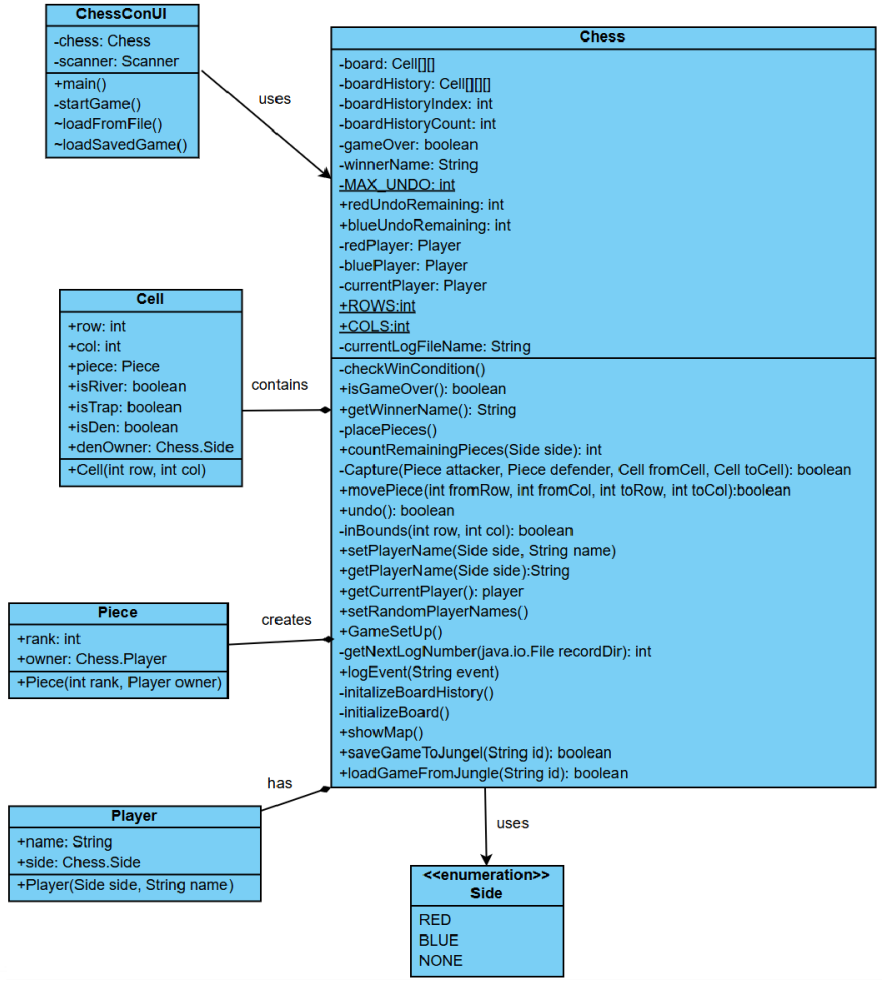
```
undos RED 1.5
```

```
Error parsing line, skipping: 'undos RED 1.5' -> For input string: "1.5"
```

```java
else if (lower.startsWith(prefix: "undos ")) {
  String[] parts = line.split(regex: "\\s+");
  if (parts.length >= 3) {
      String side = parts[1].toUpperCase();
      int val = Integer.parseInt(parts[2]);
      if ("RED".equals(side)) redUndoRemaining = Math.max(a: 0, Math.min(MAX_UNDO, val));
      else if ("BLUE".equals(side)) blueUndoRemaining = Math.max(a: 0, Math.min(MAX_UNDO, val));
  } else {
      System.err.println("Malformed undos line, skipping: " + line);
  }
```

The overall design of the game, i.e., the combination of Architecture, Structure of and relationship among main code components





One important lesson learned from the project

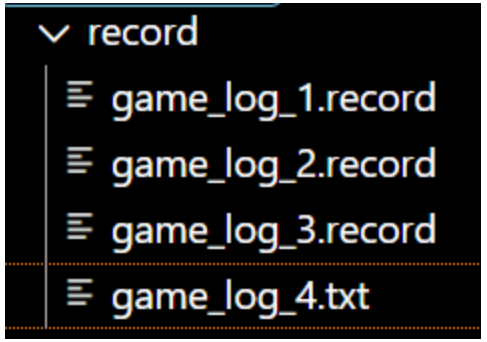
- Error handling mechanism is important

  especially for file writing

  - All files are visible to users and can be edited (no

    access restrictions)

```
Enter record number (e.g. 1, 11), or leave empty to go back to previous page: 4
Attempting to load: record\game_log_4.record
Record file not found: record\game_log_4.record
```



A txt file created manually by user.