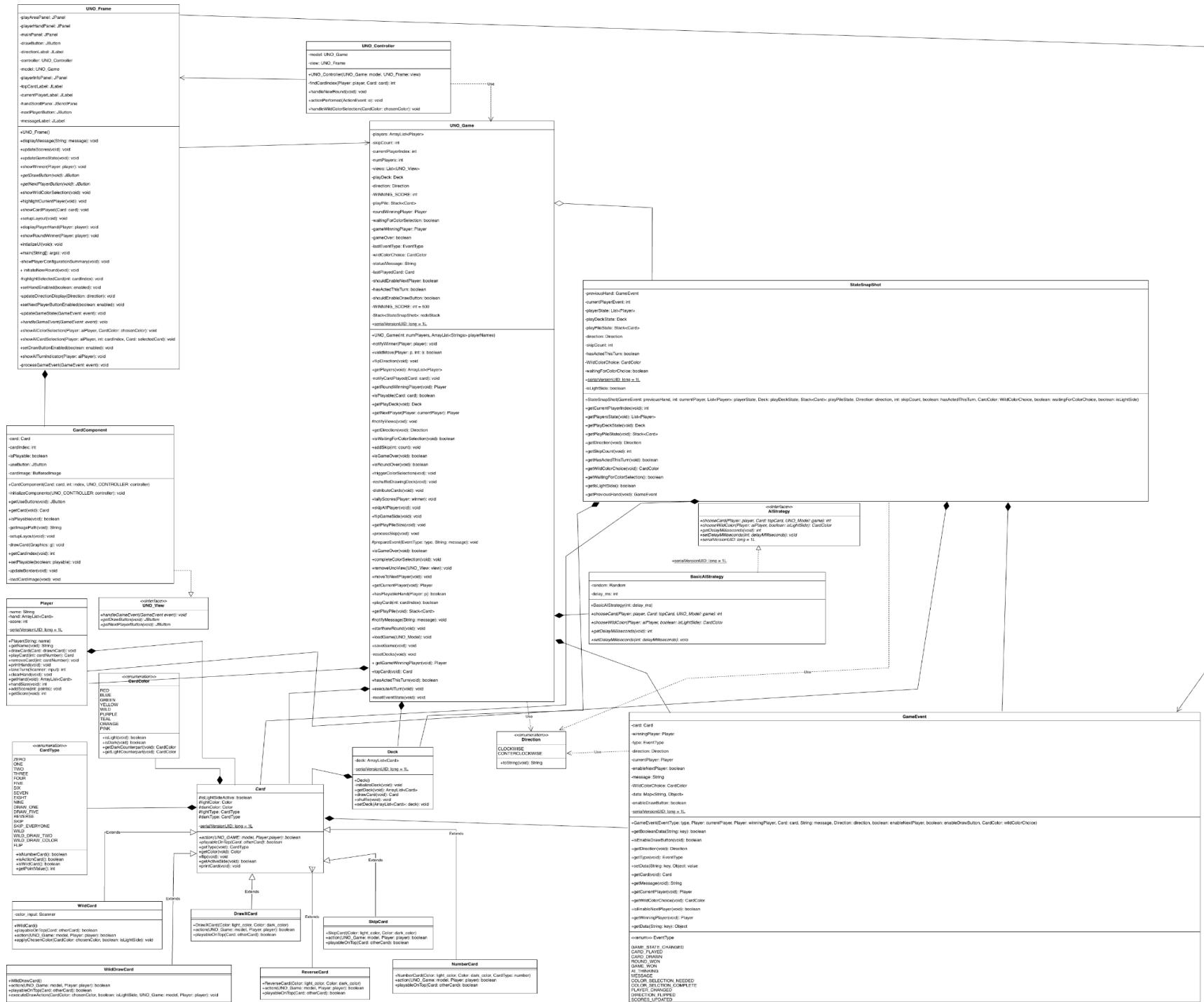
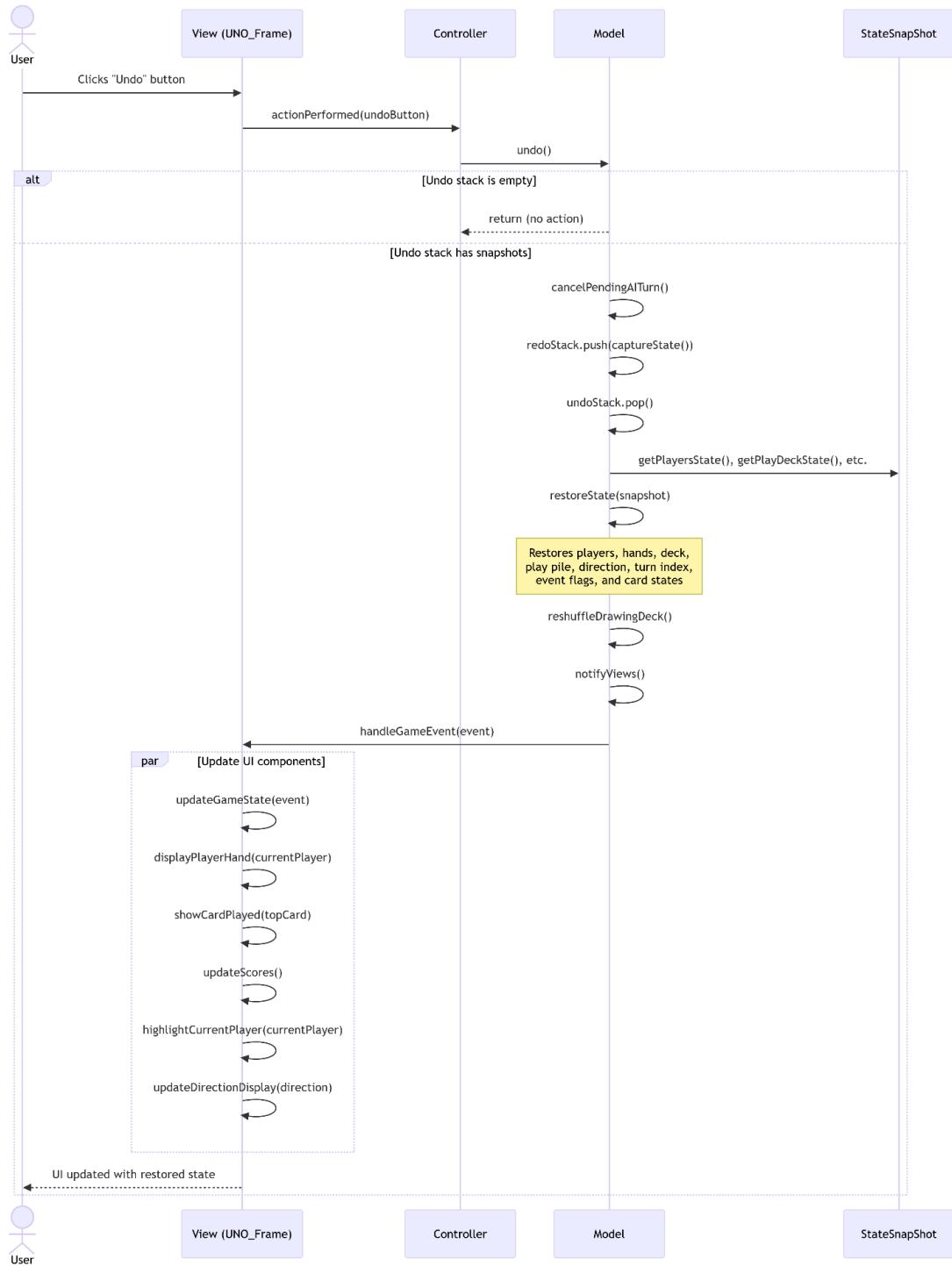


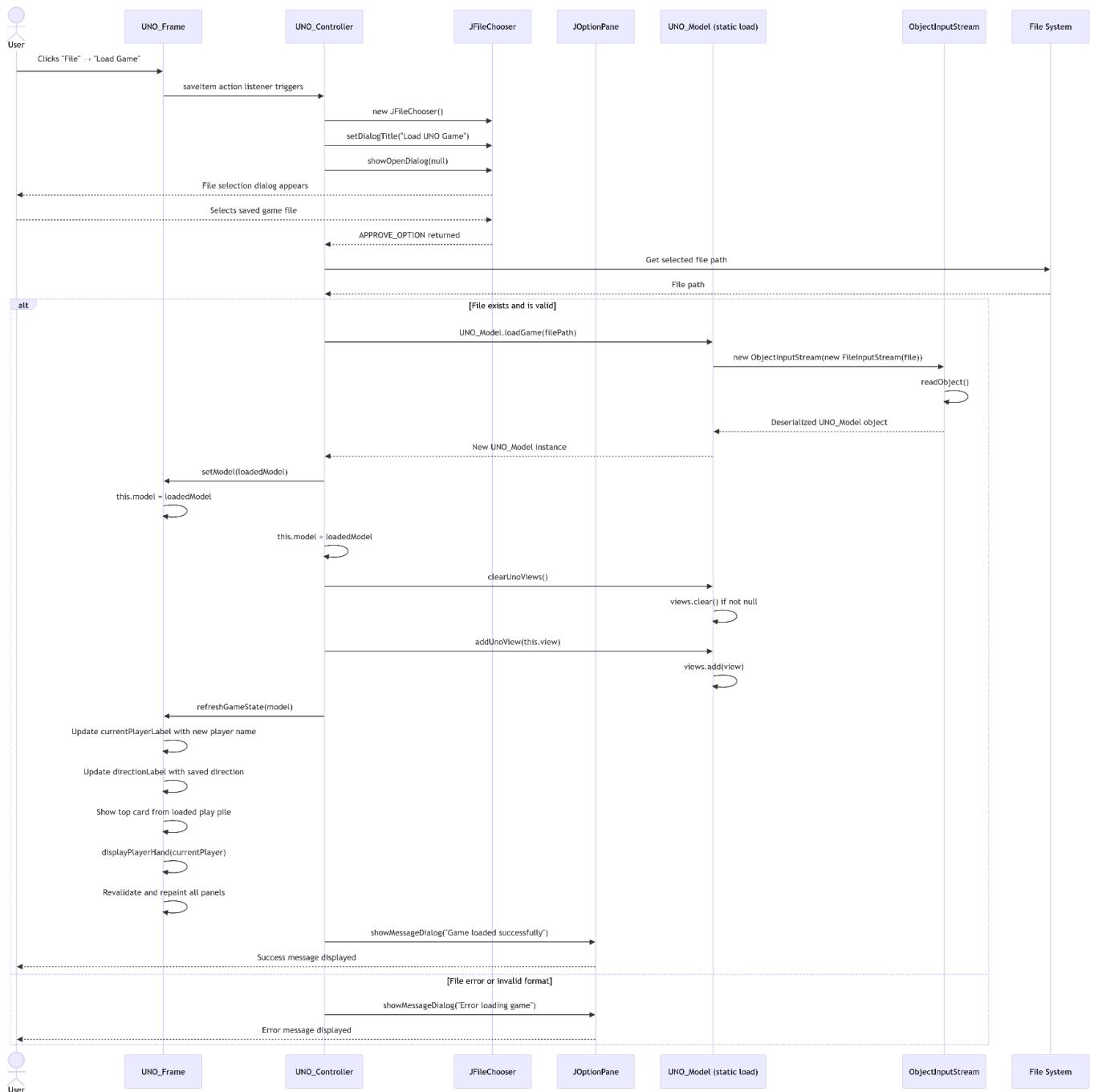
Uml Diagram:



Sequence Diagram (Undoing a Move):



Sequence Diagram (Deserializing / loading a saved game):



Data structures:

Changes made to UML and data Structures from Milestone 3:

Milestone 4 introduced some structural updates to support undo/redo, saving/loading the game, and game restart functionality

1. Added Undo/Redo System:

The undo/redo system allows the game to revert or re-apply previous states for full turnlevel history control.

- a. Introduced two main data structures in the model:
 - i. private Stack<StateSnapShot> undoStack; stores previous game states so the model can revert one step backward.
 - ii. private Stack<StateSnapShot> redoStack; stores states that were undone so they can be re-applied.
- b. UML updated to show the model maintaining associations to multiple StateSnapShot object

2. New StateSnapShot class:

This class captures a copy of the entire game state so it can be restored exactly later.

- a. Stores copies of players, deck, play pile, direction, and event state. This allows complete reconstruction of the game at any moment.
- b. Appears in UML as new Component connected to the model

3. Model Modifications

The Model was expanded to manage saving/loading state, restoring snapshots, and handling turn history.

- a. Added methods : captureState(), restoreState(), undo(), redo(), saveGame(), loadGame().
- b. Model now drives state restoration logic and supports game restarting

4. Controller Modifications

The Controller was extended to forward new UI actions (undo/redo/load/save) into model operations.

- a. Added event handling for Undo, Redo, Save Game, Load Game, and New Game buttons, connecting UI actions to model logic
- b. The controller now delegates these features to the Model.

5. Frame Modifications:

The Frame was updated to visually support the new functionality and show the game's interaction controls.

- a. Added new UI components: Undo / Redo / Save / Load / Restart buttons
- b. Updated layout to display game status and display the new buttons accordingly
- c. UML updated to include the new UI controls.

6. Serialization Support Across Classes

Serialization enables saving and restoring the complete game state in external files.

- a. Model, Snapshot, and any related classes now implement Serializable to allow full game persistence.
- b. UML updated with the «serializable» stereotype to document all classes supporting file-based state storage.