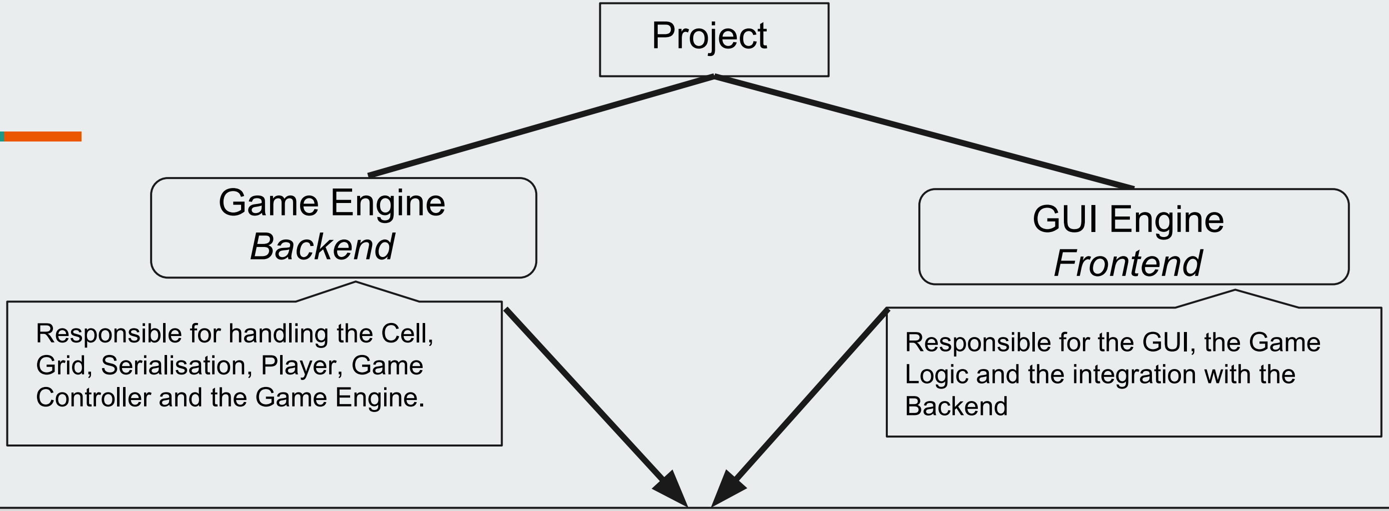
Implementation Details

Chain Reaction - Brihi Joshi, 2016142 | Aditya Chetan, 2016217



Project

Game Engine
Backend

Responsible for handling the Cell, Grid, Serialisation, Player, Game Controller and the Game Engine.

GUI Engine
Frontend

Responsible for the GUI, the Game Logic and the integration with the Backend

Design Patterns

- **Iterator** - Used in Collection items like HashSet
- **Adaptor** - Used in functions like convertGUItoGrid, ColorUtil, convertGridtiGUI, convertIndex, getIndexofStackPanefromCo-ords
- **Singleton** - GameEngine and GUIMain
- **Flyweight** - *duplicate* method
- **Decorator** - Save/Load functions and setEmptyGrid
- **Facade** - For creating separate pages, Undo and Resume states separately handled by GameController
- **Composite** - Shown in the creation of GridPanes for the GUI
- **Observer** - the GameEngine registers with GUIMain
- **State** - Used in selection of coodinates and decision of end games.

Issues/Problems faced

1. **Undo/Resume**

Problem - Edge Cases (of Unexpected exits) were not tackled and new cases kept popping up as the coding process reached completion.

How did we resolve it - Continuous testing at every stage by using Breakpoints and Print statements.

2. **Animation**

Problem - Initially only one level of animation was shown and then the final stage was displayed. Later, issues in simultaneous transitions were occurring.

How did we resolve it - Changed the structure of the code. We implemented mutual recursion to ensure each level of animation gets displayed and used ParallelTransition to parallelize the TranslateTransitions.

3. **GUI Positioning**

Problem - Since no SceneBuilder and FXML files were used, there was a lot of trouble in deciding and implementing the GUI elements.

How did we resolve it - Searched extensively about the documentation of GUI components like ColorPicker, Menu Bar, PopOver, BorderPane

4. **Deployment**

Problem - Our initial Bonus idea included making a desktop java application by created a .jar file. However, the images and other resources were not being integrated.

How did we resolve it - We tried converting our code into a Maven project and were successful in it. Finally, a succesful MacOS build has been generated as a desktop application!

Individual Contribution

Aditya Chetan

- Implemented the game logic
- Handled rotation animations
- Handled translation animations
- Handled scene transition in a single stage
- Handled the linking between the game logic and the GUI
- Created GUI components like PopOver and MenuBar

Brihi Joshi

- Implemented the game logic
- Handled Undo and Resume operations (serialization and deserialization), and the edge cases associated with them
- Handled button handlers for menu buttons.
- Created the GUIs for all the pages from scratch.
- Implemented the utility functions for assisting in game logic

Extra Features for Bonus

1. A fully functional **Desktop Application** for macOS X
2. **Color Wheel -** Giving the player an option to create their own color
3. **Sound effects -** No game is complete without sound effects. We have added sounds for a correct move, an incorrect move and the end of the game.