DEADWOOD GUI IMPLEMENTATION

This project enhances the Deadwood game with a graphical user interface (GUI)

- Baj Brar, Grey Pendras

DESIGN & IMPLEMENTATION STRATEGY

Model: Game logic (players, roles, rooms, turns, etc.)

View: Graphical representation (game board, players, cards)

Controller: Handles user interactions (clicks, upgrades, acting)

DESIGN & IMPLEMENTATION STRATEGY

we used the Observer Pattern to maintain the separation between the Model and View components, which is a key part of the Model-View-Controller (MVC) design pattern.

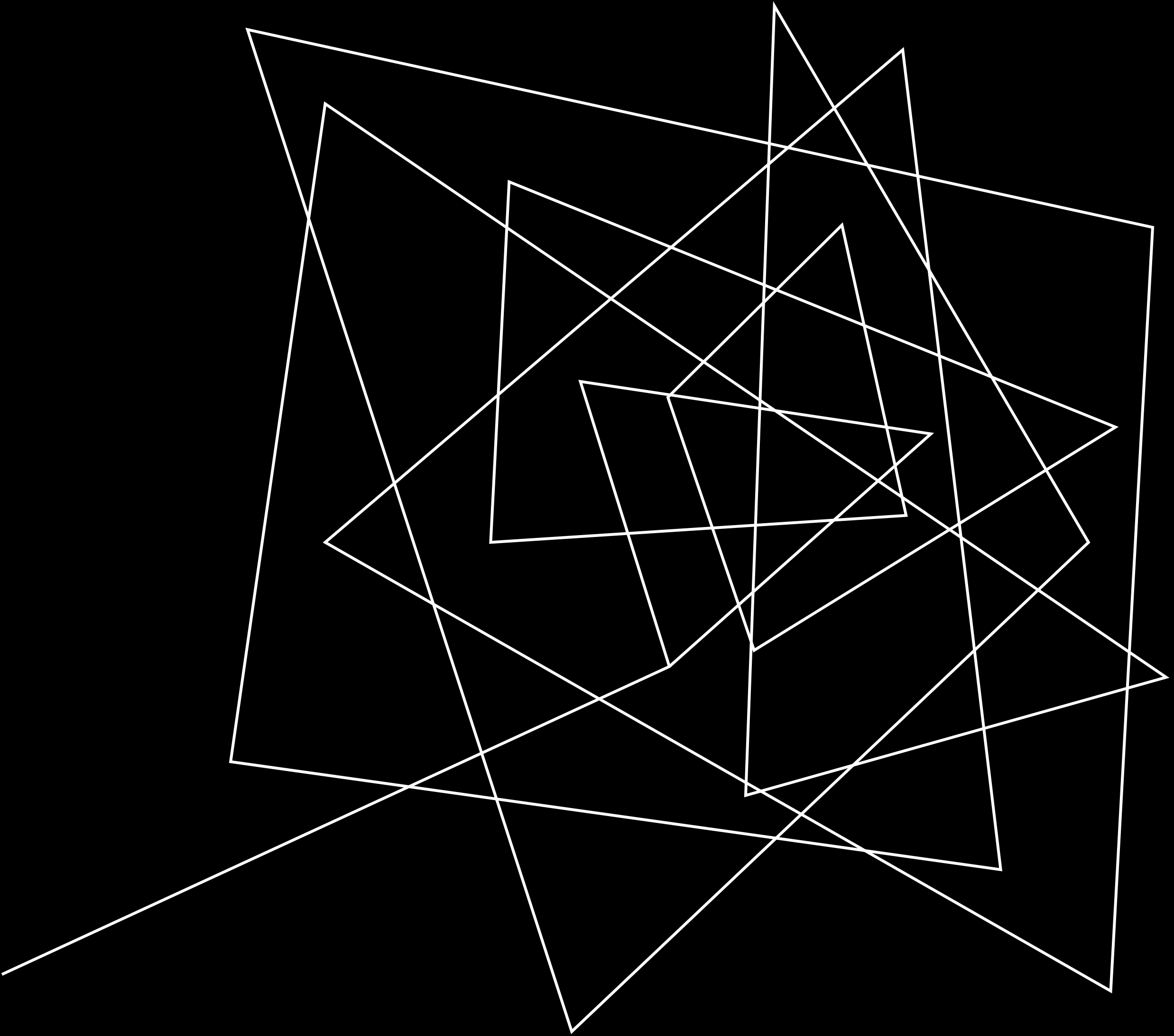
- Model as the Subject: The game’s core logic (such as players' stats, actions, and the current state of the board) is the Subject. The Model is responsible for maintaining the game state and notifying observers (the GUI) whenever an update occurs (when a player completes a turn or when the player’s position changes).

- View as the Observer: The View acts as the Observer, listening for changes in the Model. Whenever the Model changes (player stats are updated, a new scene starts, or a player moves), the View is notified and updates the GUI elements accordingly.

- We added Observer interfaces to both the Model and View. Whenever a significant change occurs in the Model, such as the player moving to a new room, the Model triggers an event that the View picks up and responds to by updating the GUI elements (moving player icons, updating stats).

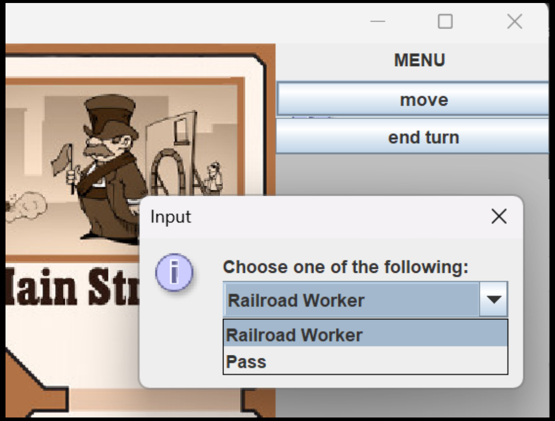
IMPLEMENTATION CHOICES & TECHNOLOGIES USED

- **Java Swing** for GUI development (lightweight and flexible).
- **Observer Pattern** to keep the View updated when the Model changes.
- **Event Listeners** for handling mouse clicks and interactions.
- **Layout Managers (BorderLayout, GridBagLayout, etc.)** for organizing UI elements.
- **Use of Images** for visual elements (players, cards, scene backgrounds), (These were provided)



GUI FEATURES

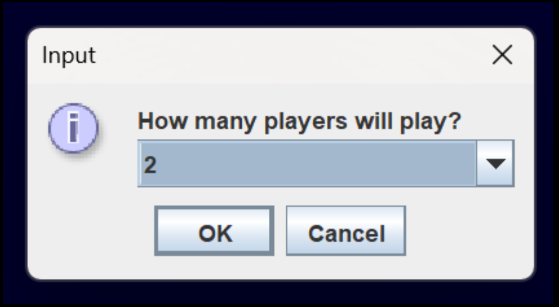
Through Java Swing we were able to create buttons and menus to prompt the user with options. These options were limited to force the user to make limited valid moves and not allow invalid ones.



Choose one of the following:

Railroad Worker

Railroad Worker
Pass



How many players will play?

2

STRUGGLES WITH JAVA SWING/GUI

Dynamically Keeping Track of Players in the GUI:

- A big challenge was keeping the players' positions and states up to date in real time. Since players can move between rooms, change actions, and have different statuses, displaying this information accurately and dynamically in the GUI was tricky.
- We struggled to manage and update each player's graphical representation (avatars, status indicators) on the game board without constant redraws or performance issues.

Displaying Dynamic Elements (e.g., Scene Wrapping and Player Movement):

- Another challenge was making sure that actions like scene wrapping, player movements, and role acting were visually represented in a way that made sense and felt intuitive.
- We struggled with keeping track of dynamic elements like the remaining shots in a scene or updating player stats like credits, rehearsals, and rank after every turn.