Manual Test Plan

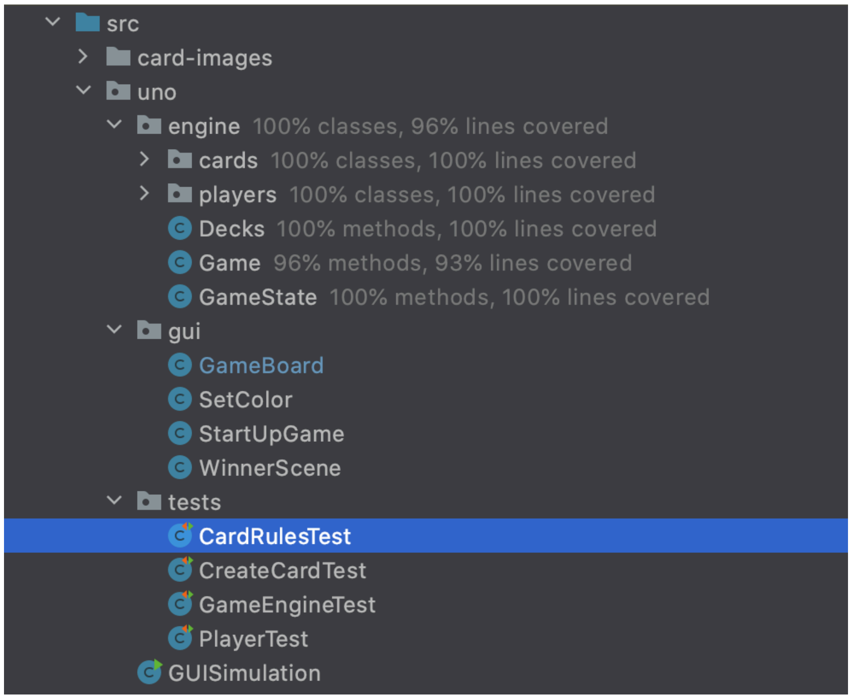
Author: Pranav Narahari

Prerequisites:

- Tools and Version:
    - IntelliJ IDEA 2020.3.2
    - Dependencies:
        - Java SDK version 13.0.1
        - J-unit (junit-4.13)
        - hamcrest-2.2

Environment Setup and Configurations:

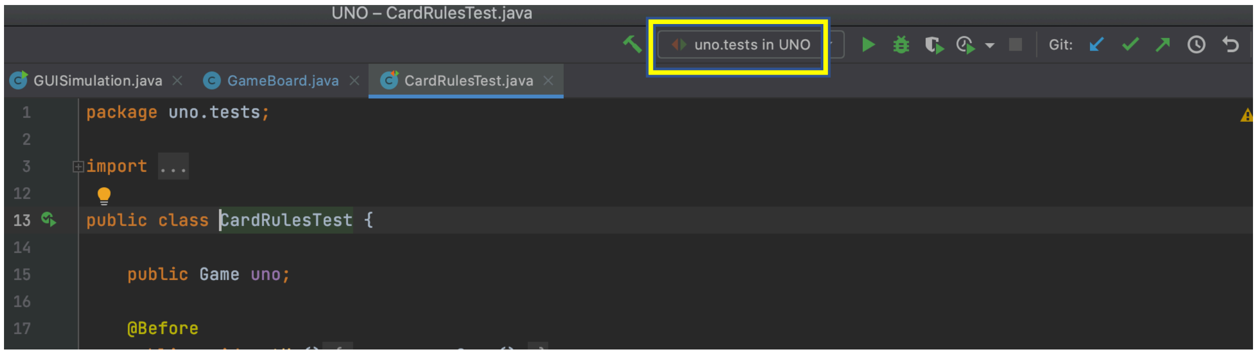
- **Project Organization:**



```
src
  card-images
  uno
    engine  100% classes, 96% lines covered
      cards  100% classes, 100% lines covered
      players  100% classes, 100% lines covered
      Decks  100% methods, 100% lines covered
      Game  96% methods, 93% lines covered
      GameState  100% methods, 100% lines covered
    gui
      GameBoard
      SetColor
      StartUpGame
      WinnerScene
    tests
      CardRulesTest
      CreateCardTest
      GameEngineTest
      PlayerTest
    GUISimulation
```

- Project uses the Model-View-Controller design framework. The model elements of the project are located within the engine package, the view elements are located in the gui package, and the controller file is GUISimulation.java.
- Additionally, the tests are in the tests package.

- **Unit-testing:**
    - The uno.tests should be the configuration to run to run all tests.



UNO – CardRulesTest.java

uno.tests in UNO

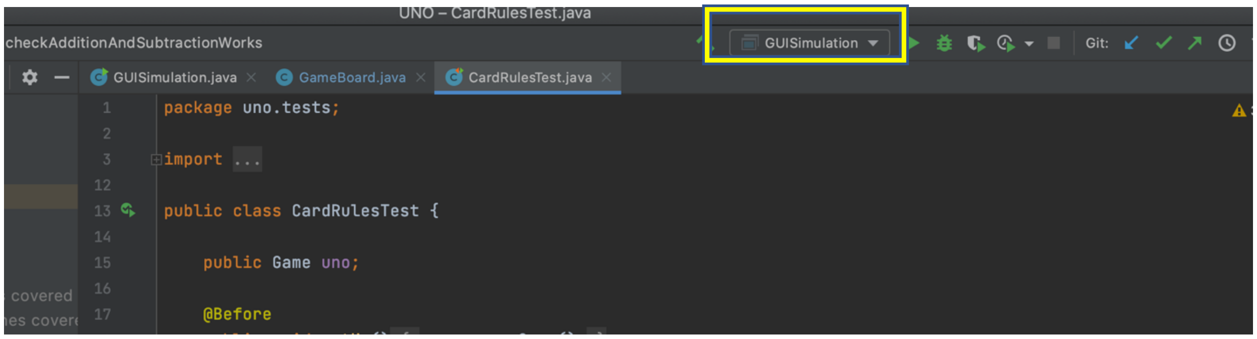
Git:

GUISimulation.java | GameBoard.java | CardRulesTest.java

```
package uno.tests;

import ...

public class CardRulesTest {

    public Game uno;

    @Before
```

    - to test specifically Player classes, run with the configuration PlayerTest.
    - to test specifically Card Rules functionality, run with the configuration CardRulesTest.
    - to test the Game as a whole, run with the configuration GameEngineTest.
    - to test the Card classes, run the configuration CreateCardTest.

- **Game Simulation:**
    - The GUISimulation configuration should be run.



UNO – CardRulesTest.java

checkAdditionAndSubtractionWorks

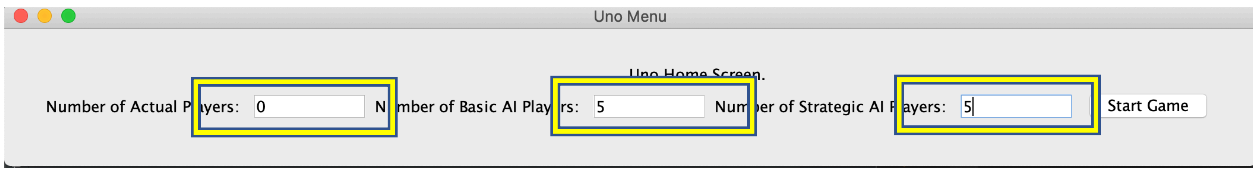
GUISimulation

Git:

GUISimulation.java | GameBoard.java | CardRulesTest.java

```
package uno.tests;

import ...

public class CardRulesTest {

    public Game uno;

    @Before
```

- **Testing via Game Simulation:**
    - set all players to AI players and you will be able to run through all the game states without as much interaction. This will be helpful for efficient testing that can't be covered in unit tests.



Uno Menu

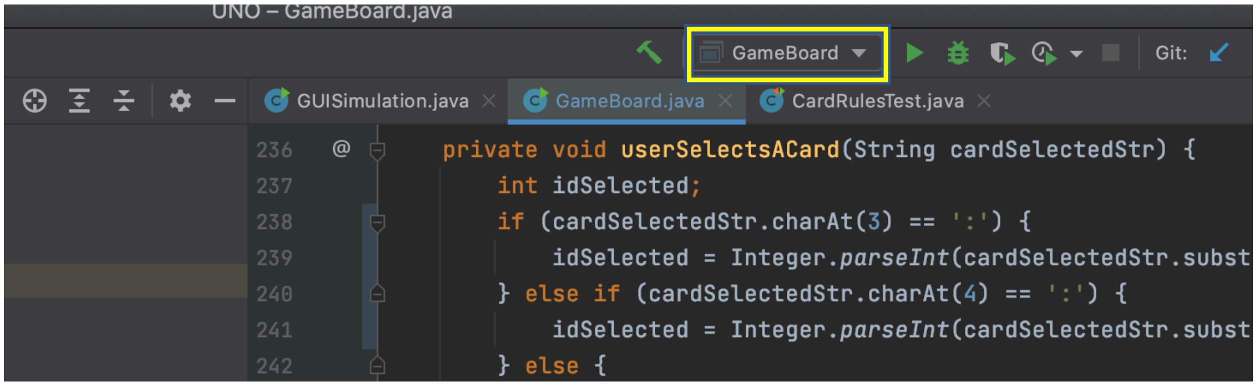
Uno Home Screen.

Number of Actual Players: 0 Number of Basic AI Players: 5 Number of Strategic AI Players: 5 Start Game

- **Testing individual views:**

- Run the main file for each of the views to get an understanding of what each view does on it's own.

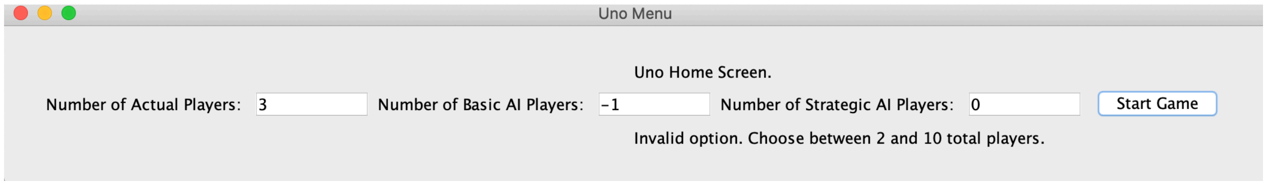
- **Example: Run GameBoard**



```
private void userSelectsACard(String cardSelectedStr) {
    int idSelected;
    if (cardSelectedStr.charAt(3) == ':') {
        idSelected = Integer.parseInt(cardSelectedStr.subst
    } else if (cardSelectedStr.charAt(4) == ':') {
        idSelected = Integer.parseInt(cardSelectedStr.subst
    } else {
```

Operations and the results:

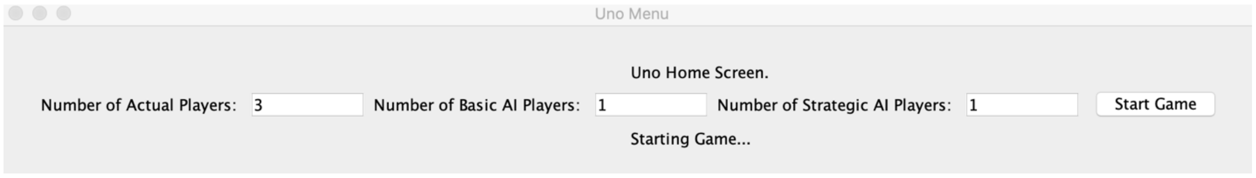
- Operations:
  - **Set number of players:**
    - A valid inserted number of human players and AI and clicking start game results in the game beginning with Player 1's turn.
    - An invalid number of players results in the user having to re-enter the number of players until they choose a valid number of players.
    - The file for this view is StartUpGame.java


Uno Menu

Uno Home Screen.

Number of Actual Players: 3 Number of Basic AI Players: -1 Number of Strategic AI Players: 0 Start Game

Invalid option. Choose between 2 and 10 total players.

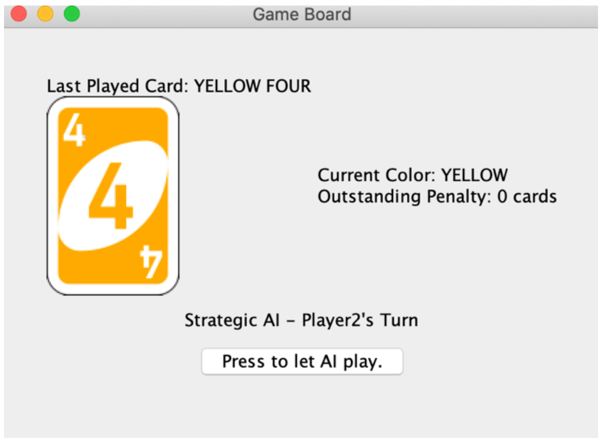

Uno Menu

Uno Home Screen.

Number of Actual Players: 3 Number of Basic AI Players: 1 Number of Strategic AI Players: 1 Start Game

Starting Game...

  - **Playing Turn:** AI GUI.
    - Strategic and Basic AI:
      - Some user must still click button to allow AI to play
        - Note: this button was added purely for game play feedback from the AI and could be easily removed.

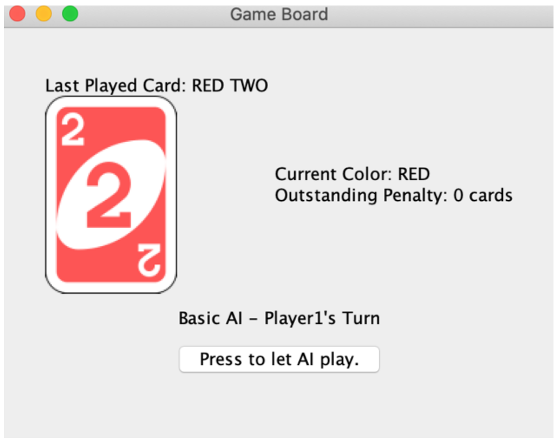

Game Board

Last Played Card: YELLOW FOUR

Current Color: YELLOW
Outstanding Penalty: 0 cards

Strategic AI – Player2's Turn

Press to let AI play.


Game Board

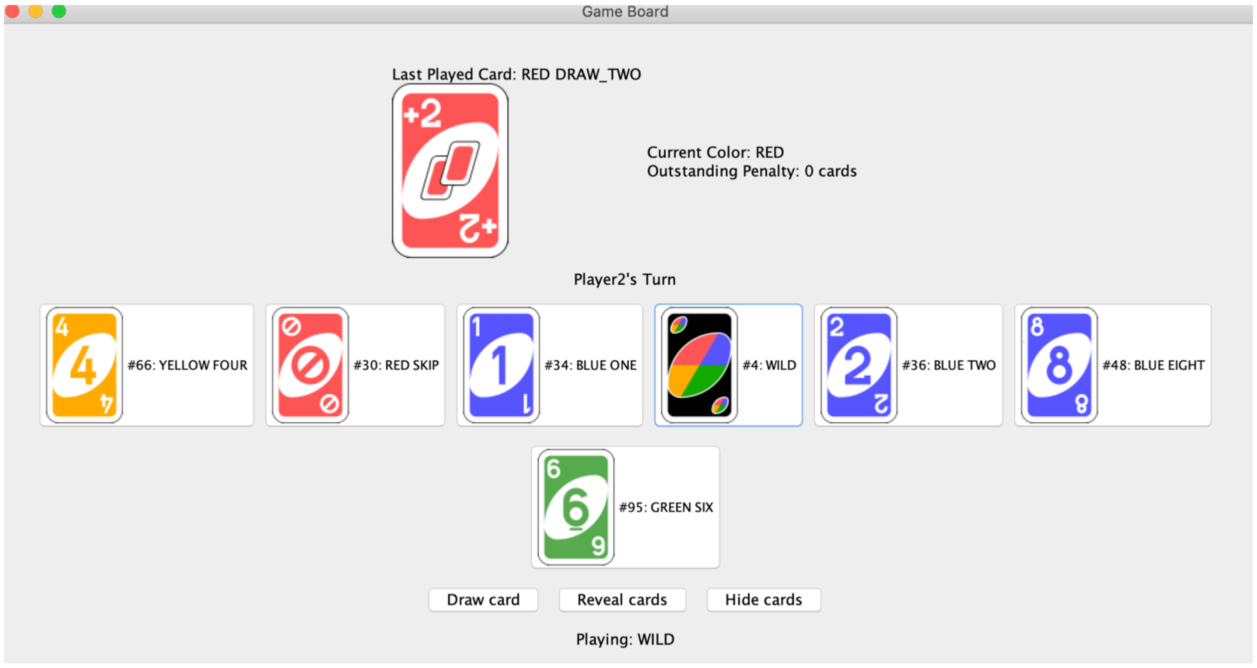
Last Played Card: RED TWO

Current Color: RED
Outstanding Penalty: 0 cards

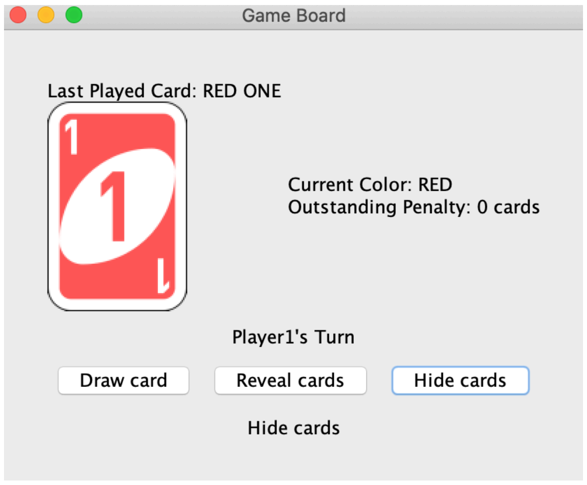
Basic AI – Player1's Turn

Press to let AI play.

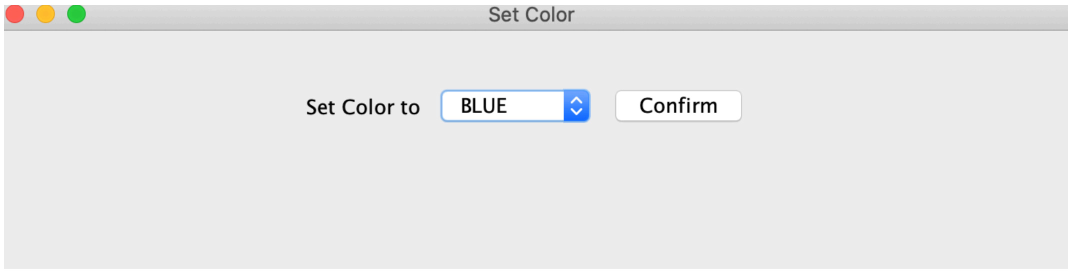
- **Playing Turn:** Non-AI GUI.



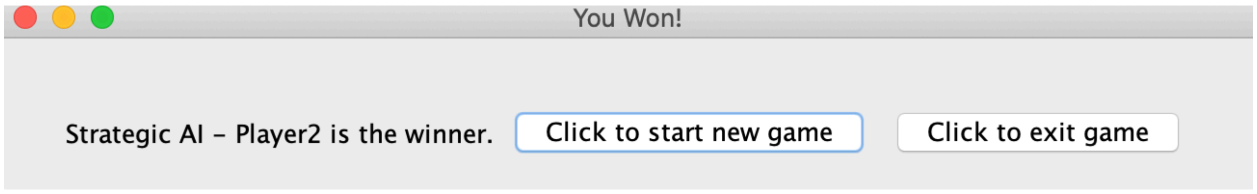
- A valid selection results in the GUI automatically going to the next player
- An invalid selection results in the user's turn not changing until they make a valid selection.
- Each card under the player name is clickable, however, it will only move on to the next player's turn if the card is playable.
- Draw card is always an option is the player would like to draw card instead of playing.
- Reveal and hide cards are buttons that update the UI to hide or show the player's cards.
- The file for this view is GameBoard.java



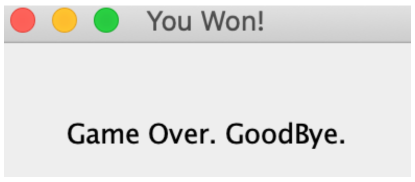
- **Choose a color:**
    - A valid selection results in the console outputting the selected color and game continuing



- **Start a new game:**
    - A valid selection results in the console prompting the user for the number of players to start a new game or ending the game.
    - “Click to start new game” takes user back to the set number of players menu.
    - “Click to exit game” ends the game and shows ending screen.



Strategic AI – Player2 is the winner.



Game Over. GoodBye.