# JAVATAIRE

Implementation Manual

**Class Card:**

- Each Card object is initialized using its constructor to give it a suit and value.
- Getters and setters are in place for suit and value, as well as for its color based on suit.
- getName(); returns a String of the Card's name based on its suit and value. For use in loading images.
- isFlipped(); returns a Boolean stating whether the Card is flipped (visible).
- isSelected(); returns a Boolean stating whether the Card is selected by the user.
- flip(); flips the Card accordingly.
- select(); selects the Card accordingly.

**Class Deck:**

- Manages Card and stores Card objects. Creates piles for stock & waste, as well as lists of stacks of Cards for the tableau and foundations.
- initialize(); creates a standard deck of Cards.
- swap(int, int); used for shuffling Cards.
- shuffle(); randomizes the order of each Card in the deck.
- createTableau(); places tableaus of Cards in default starting positions.
- placeDown(); returns Card from top of the stock.
- popStock(); takes card at top of the stock and places it into waste.
- reset(); restocks waste cards.
- checkFoundationMove(Card, Card); checks if Card can be moved into the foundation at target Card. Returns Boolean (true if it can be moved).
- checkTableauMove(Card, Card); checks if Card can be moved into the tableau at target Card. Returns Boolean (true if it can be moved).
- isWon(); checks if win condition has been met. Boolean value true returned if game is won.
- moveCards(stack<Card>); moves a Card from one stack to another.
- selectCard(Card); sets Card's status to selected.
- deselectCards(); sets all currently selected Cards back to deselected (isSelected() == false).

**enum Suit:**

- For Card objects.

- Assigns one of the four suits to a Card.

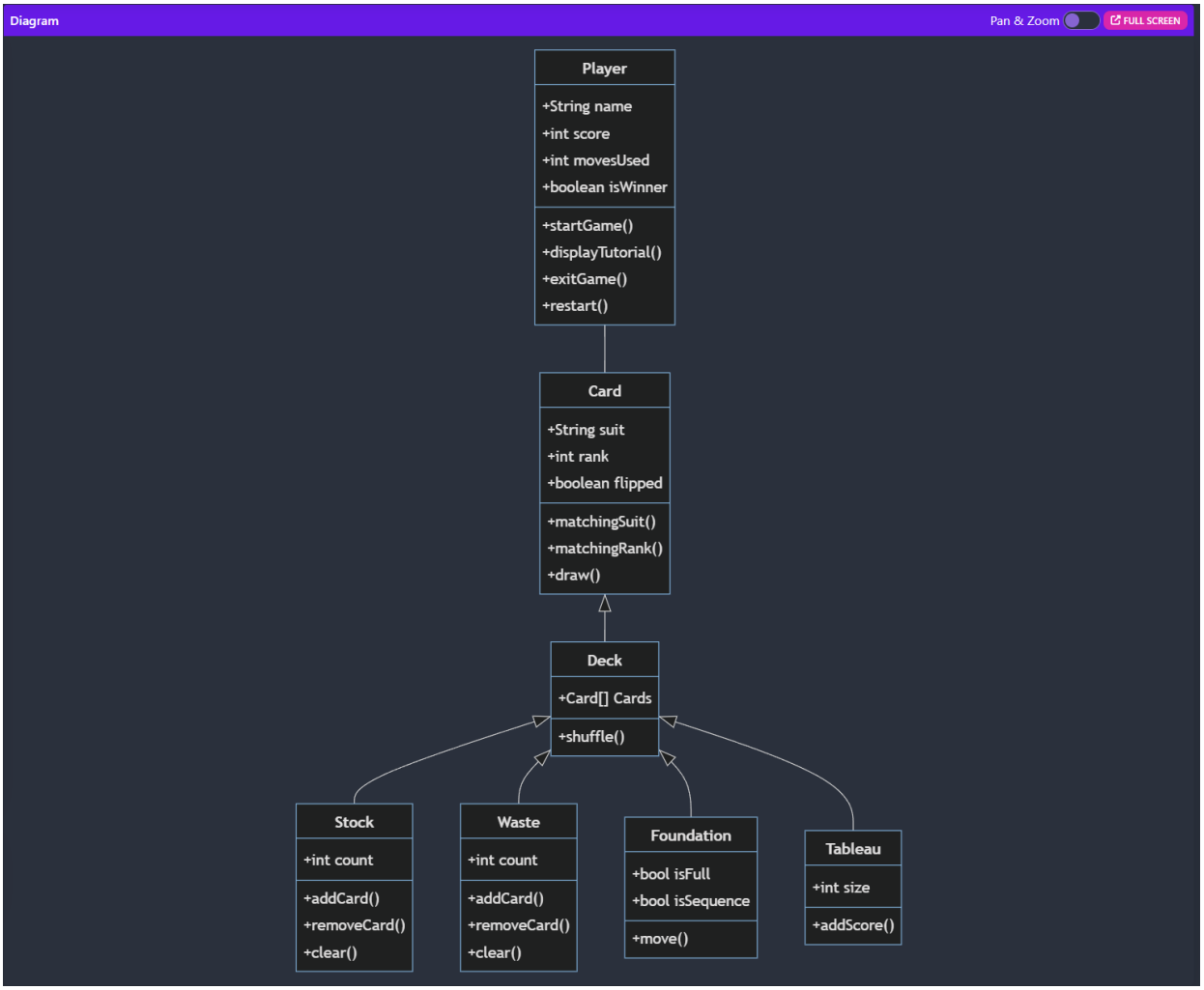
**enum Value:**

- For Card objects.
- Assigns a weight to each card depending on its value (aces low, 2-10, then jack – king).

**enum Color:**

- For Card objects.
- Red or black. For use in comparing Cards.

Initial UML for this application:




Much has changed since the original proposal for this project. After referencing many other similar projects and programs that implemented decks of cards, it made more sense to implement the elements of solitaire into a Deck class full of Card objects, as opposed to having a master Player class that holds Card objects. In my original approach, the scope of solitaire was in the wrong place. The code in the final build of the application gets rid of the Player class entirely and instead opts for the JavaFX controls to instead integrate Deck and Card as if they were any other data type.