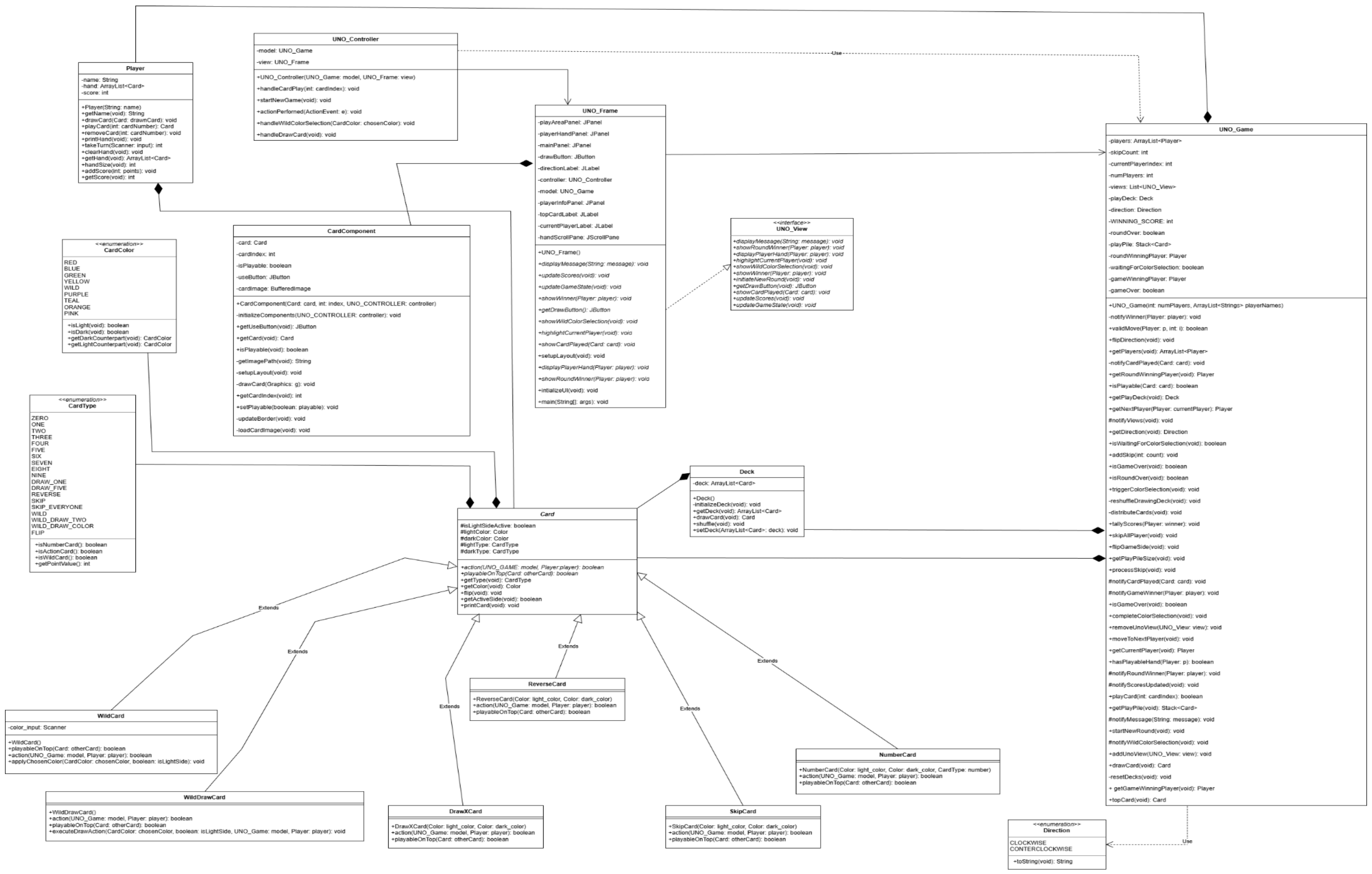
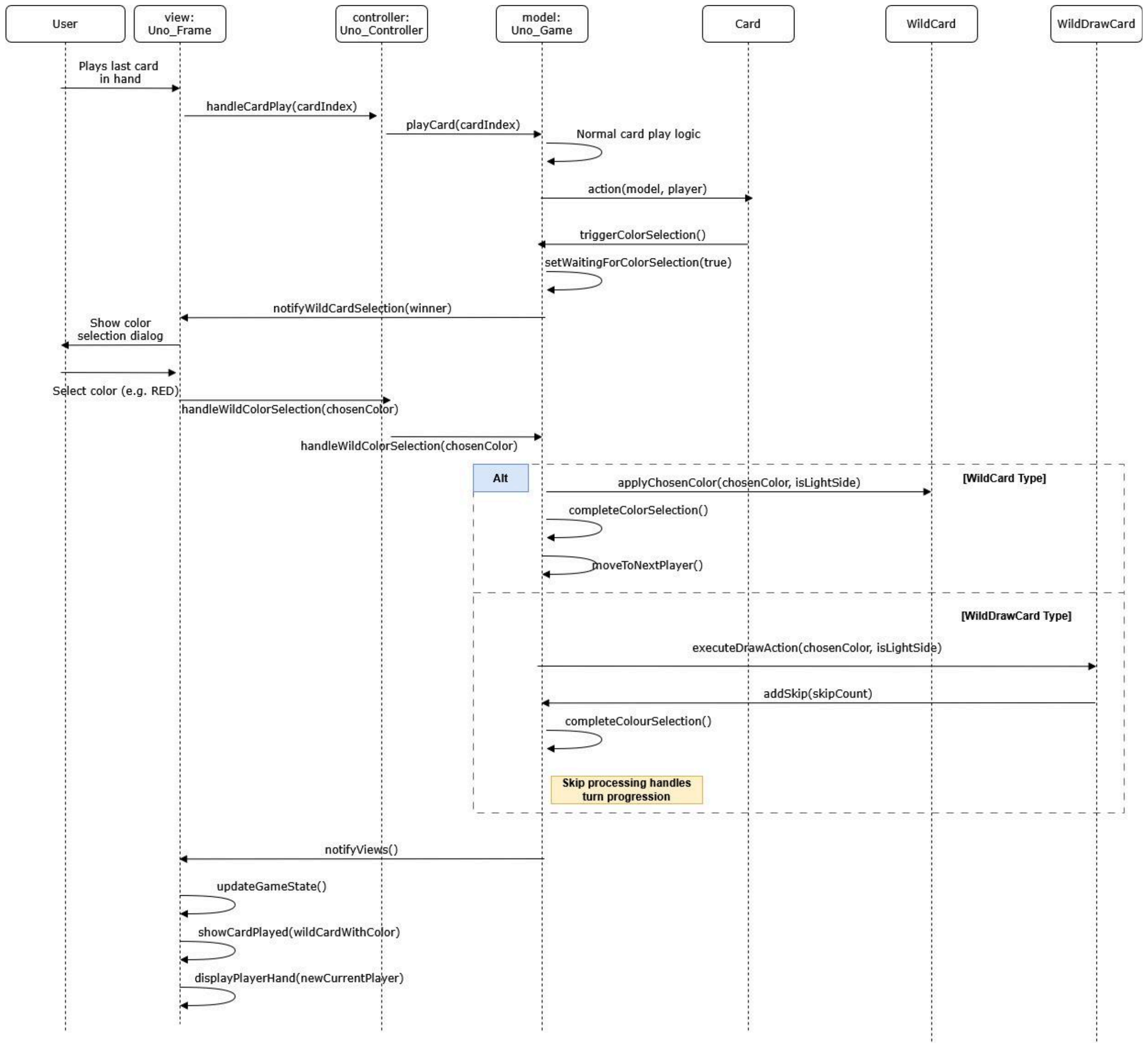
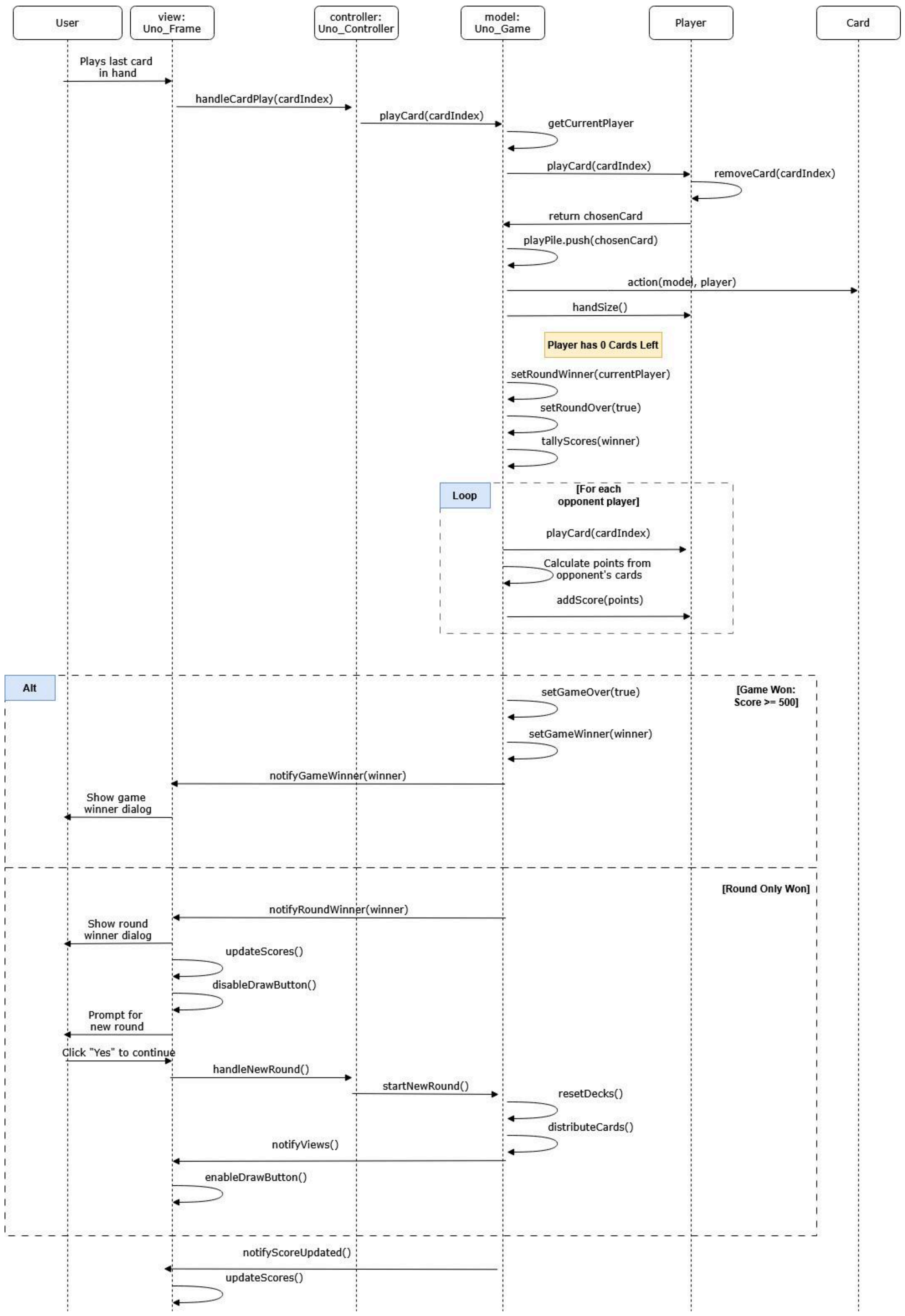
Uml Diagram:


Player
-name: String
-hand: ArrayList<Card>
-score: int
+Player(String: name)
+getName(void): String
+drawCard(Card: drawnCard): void
+playCard(int: cardNumber): Card
+removeCard(int: cardNumber): void
+printHand(void): void
+takeTurn(Scanner: input): int
+clearHand(void): void
+getHand(void): ArrayList<Card>
+handSize(void): int
+addScore(int: points): void
+getScore(void): int
UNO_Controller
-model: UNO_Game
-view: UNO_Frame
+UNO_Controller(UNO_Game: model, UNO_Frame: view)
+handleCardPlay(int: cardIndex): void
+startNewGame(void): void
+actionPerformed(ActionEvent: e): void
+handleWildColorSelection(CardColor: chosenColor): void
+handleDrawCard(void): void
Use
UNO_Frame
-playAreaPanel: JPanel
-playerHandPanel: JPanel
-mainPanel: JPanel
-drawButton: JButton
-directionLabel: JLabel
-controller: UNO_Controller
-model: UNO_Game
-playerInfoPanel: JPanel
-topCardLabel: JLabel
-currentPlayerLabel: JLabel
-handScrollPane: JScrollPane
+UNO_Frame()
+displayMessage(String: message): void
+updateScores(void): void
+updateGameState(void): void
+showWinner(Player: player): void
+getDrawButton(): JButton
+showWildColorSelection(void): void
+highlightCurrentPlayer(void): void
+showCardPlayed(Card: card): void
+setupLayout(void): void
+displayPlayerHand(Player: player): void
+showRoundWinner(Player: player): void
+intializeUI(void): void
+main(String[]: args): void
<<interface>>
UNO_View
+displayMessage(String: message): void
+showRoundWinner(Player: player): void
+displayPlayerHand(Player: player): void
+highlightCurrentPlayer(void): void
+showWildColorSelection(void): void
+showWinner(Player: player): void
+initiateNewRound(void): void
+getDrawButton(void): JButton
+showCardPlayed(Card: card): void
+updateScores(void): void
+updateGameState(void): void
UNO_Game
-players: ArrayList<Player>
-skipCount: int
-currentPlayerIndex: int
-numPlayers: int
-views: List<UNO_View>
-playDeck: Deck
-direction: Direction
-WINNING_SCORE: int
-roundOver: boolean
-playPile: Stack<Card>
-roundWinningPlayer: Player
-waitingForColorSelection: boolean
-gameWinningPlayer: Player
-gameOver: boolean
+UNO_Game(int: numPlayers, ArrayList<Strings> playerNames)
-notifyWinner(Player: player): void
+validMove(Player: p, int: i): boolean
+flipDirection(void): void
+getPlayers(void): ArrayList<Player>
-notifyCardPlayed(Card: card): void
+getRoundWinningPlayer(void): Player
+isPlayable(Card: card): boolean
+getPlayDeck(void): Deck
+getNextPlayer(Player: currentPlayer): Player
#notifyViews(void): void
+getDirection(void): Direction
+isWaitingForColorSelection(void): boolean
+addSkip(int: count): void
+isGameOver(void): boolean
+isRoundOver(void): boolean
+triggerColorSelection(void): void
-reshuffleDrawingDeck(void): void
-distributeCards(void): void
+tallyScores(Player: winner): void
+skipAllPlayer(void): void
+flipGameSide(void): void
+getPlayPileSize(void): void
+processSkip(void): void
#notifyCardPlayed(Card: card): void
#notifyGameWinner(Player: player): void
+isGameOver(void): boolean
+completeColorSelection(void): void
+removeUnoView(UNO_View: view): void
+moveToNextPlayer(void): void
+getCurrentPlayer(void): Player
+hasPlayableHand(Player: p): boolean
#notifyRoundWinner(Player: player): void
#notifyScoresUpdated(void): void
+playCard(int: cardIndex): boolean
+getPlayPile(void): Stack<Card>
#notifyMessage(String: message): void
+startNewRound(void): void
#notifyWildColorSelection(void): void
+addUnoView(UNO_View: view): void
+drawCard(void): Card
-resetDecks(void): void
+ getGameWinningPlayer(void): Player
+topCard(void): Card
<<enumeration>>
CardColor
RED
BLUE
GREEN
YELLOW
WILD
PURPLE
TEAL
ORANGE
PINK
+isLight(void): boolean
+isDark(void): boolean
+getDarkCounterpart(void): CardColor
+getLightCounterpart(void): CardColor
CardComponent
-card: Card
-cardIndex: int
-isPlayable: boolean
-useButton: JButton
-cardImage: BufferedImage
+CardComponent(Card: card, int: index, UNO_CONTROLLER: controller)
-initializeComponents(UNO_CONTROLLER: controller): void
+getUseButton(void): JButton
+getCard(void): Card
+isPlayable(void): boolean
-getImagePath(void): String
-setupLayout(void): void
-drawCard(Graphics: g): void
+getCardIndex(void): int
+setPlayable(boolean: playable): void
-updateBorder(void): void
-loadCardImage(void): void
<<enumeration>>
CardType
ZERO
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
DRAW_ONE
DRAW_FIVE
REVERSE
SKIP
SKIP_EVERYONE
WILD
WILD_DRAW_TWO
WILD_DRAW_COLOR
FLIP
+isNumberCard(): boolean
+isActionCard(): boolean
+isWildCard(): boolean
+getPointValue(): int
Deck
-deck: ArrayList<Card>
+Deck()
-initializeDeck(void): void
+getDeck(void): ArrayList<Card>
+drawCard(void): Card
+shuffle(void): void
+setDeck(ArrayList<Card>: deck): void
Card
#isLightSideActive: boolean
#lightColor: Color
#darkColor: Color
#lightType: CardType
#darkType: CardType
+action(UNO_GAME: model, Player:player): boolean
+playableOnTop(Card: otherCard): boolean
+getType(void): CardType
+getColor(void): Color
+flip(void): void
+getActiveSide(void): boolean
+printCard(void): void
Extends
Extends
Extends
Extends
Extends
Extends
Extends
WildCard
-color_input: Scanner
+WildCard()
+playableOnTop(Card: otherCard): boolean
+action(UNO_Game: model, Player: player): boolean
+applyChosenColor(CardColor: chosenColor, boolean: isLightSide): void
ReverseCard
+ReverseCard(Color: light_color, Color: dark_color)
+action(UNO_Game: model, Player: player): boolean
+playableOnTop(Card: otherCard): boolean
NumberCard
+NumberCard(Color: light_color, Color: dark_color, CardType: number)
+action(UNO_Game: model, Player: player): boolean
+playableOnTop(Card: otherCard): boolean
WildDrawCard
+WildDrawCard()
+action(UNO_Game: model, Player: player): boolean
+playableOnTop(Card: otherCard): boolean
+executeDrawAction(CardColor: chosenColor, boolean: isLightSide, UNO_Game: model, Player: player): void
DrawXCard
+DrawXCard(Color: light_color, Color: dark_color)
+action(UNO_Game: model, Player: player): boolean
+playableOnTop(Card: otherCard): boolean
SkipCard
+SkipCard(Color: light_color, Color: dark_color)
+action(UNO_Game: model, Player: player): boolean
+playableOnTop(Card: otherCard): boolean
<<enumeration>>
Direction
CLOCKWISE
CONTERCLOCKWISE
+toString(void): String
Use

Sequence Diagram: Playing cards (e.g. WildCard and WildDrawCard)


User
view: Uno_Frame
controller: Uno_Controller
model: Uno_Game
Card
WildCard
WildDrawCard
Plays last card in hand
handleCardPlay(cardIndex)
playCard(cardIndex)
Normal card play logic
action(model, player)
triggerColorSelection()
setWaitingForColorSelection(true)
notifyWildCardSelection(winner)
Show color selection dialog
Select color (e.g. RED)
handleWildColorSelection(chosenColor)
handleWildColorSelection(chosenColor)
Alt
[WildCard Type]
applyChosenColor(chosenColor, isLightSide)
completeColorSelection()
moveToNextPlayer()
[WildDrawCard Type]
executeDrawAction(chosenColor, isLightSide)
addSkip(skipCount)
completeColourSelection()
Skip processing handles turn progression
notifyViews()
updateGameState()
showCardPlayed(wildCardWithColor)
displayPlayerHand(newCurrentPlayer)

Sequence Diagram: Winning Round and Game Handling



Data structures:

ArrayList:

Used in:
-Deck (private ArrayList<Card> deck;)
-UNO_GAME(private ArrayList<Card> drawPile;)
-Player (private ArrayList<Card> hand;)

Why:
-Has a dynamic size
-Can store objects
-able to easily access indexes
-Able to easily remove objects

Enum:

Used in:
-Color (RED, BLUE, YELLOW, etc.)
-CardType (DRAW_ONE, WILD, etc.)
-Direction (CLOCKWISE or COUNTERCLOCKWISE)

Why:
-It's a fixed set of constants
-Prevents invalid values
-Has built in helper functions

Stack:

Used in:
-UNO_GAME (private Stack<Card> playPile;)

Why:
-The Uno play pile is LIFO since the most recently played card is at the top

Changes made to UML and data structures from Milestone 1:

The project now follows the MVC (Model-View-Controller) architecture. The updated UML diagram includes the MVC-based implementation of UNO_Game, UNO_View (and UNO_Frame), and UNO_Controller, along with the newly introduced Card Component class. Several methods from Milestone 1 have been revised and reorganized to align with this structure. The core game logic has been transferred from UNO_Game to the Model, establishing clear relationships among the model, view, and controller to create a more modular, maintainable, and scalable system.