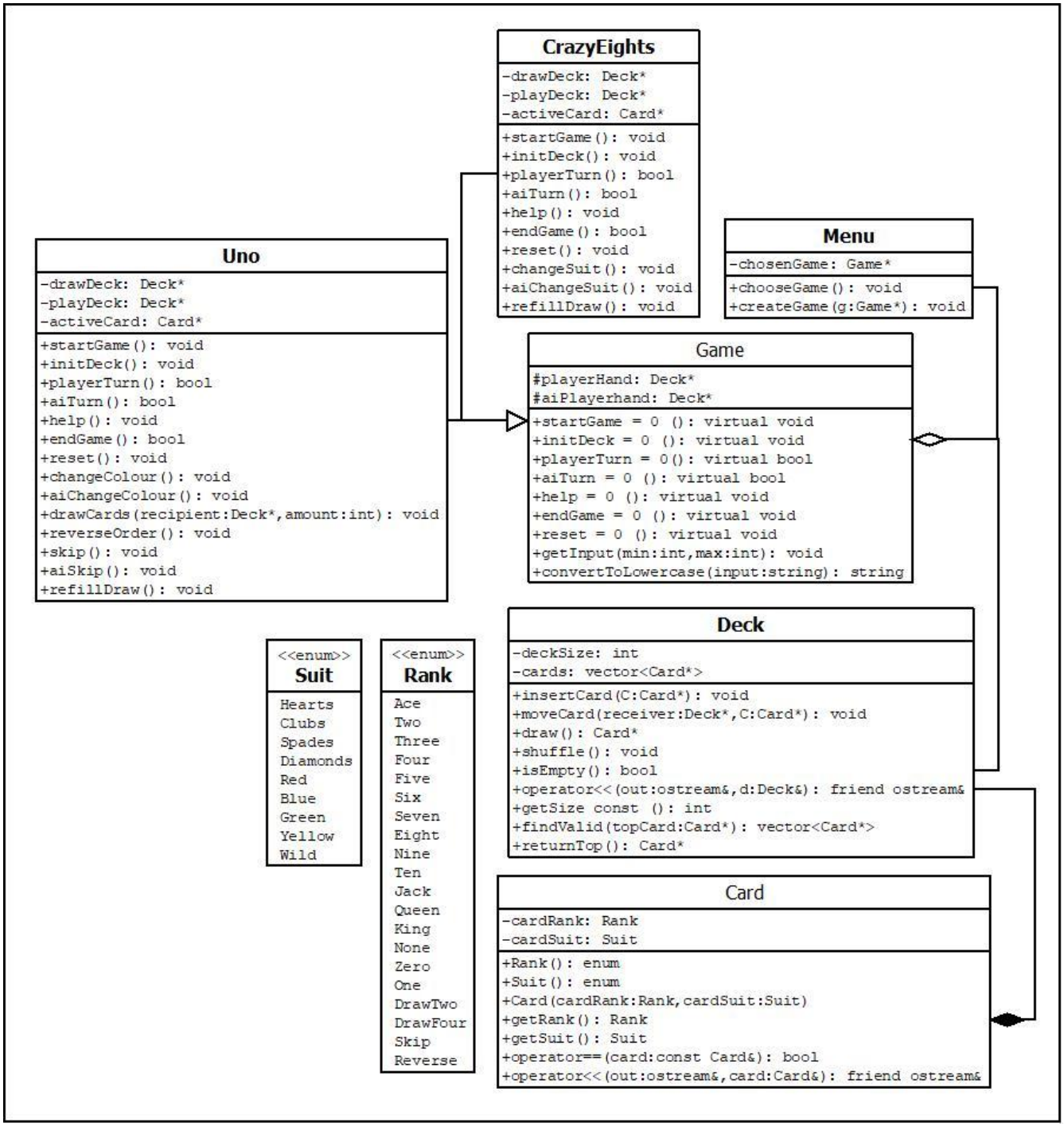
UML Diagram:


CrazyEights
-drawDeck: Deck*
-playDeck: Deck*
-activeCard: Card*
+startGame(): void
+initDeck(): void
+playerTurn(): bool
+aiTurn(): bool
+help(): void
+endGame(): bool
+reset(): void
+changeSuit(): void
+aiChangeSuit(): void
+refillDraw(): void
Menu
-chosenGame: Game*
+chooseGame(): void
+createGame(g:Game*): void
Uno
-drawDeck: Deck*
-playDeck: Deck*
-activeCard: Card*
+startGame(): void
+initDeck(): void
+playerTurn(): bool
+aiTurn(): bool
+help(): void
+endGame(): bool
+reset(): void
+changeColour(): void
+aiChangeColour(): void
+drawCards(recipient:Deck*,amount:int): void
+reverseOrder(): void
+skip(): void
+aiSkip(): void
+refillDraw(): void
Game
#playerHand: Deck*
#aiPlayerhand: Deck*
+startGame = 0 (): virtual void
+initDeck = 0 (): virtual void
+playerTurn = 0(): virtual bool
+aiTurn = 0 (): virtual bool
+help = 0 (): virtual void
+endGame = 0 (): virtual void
+reset = 0 (): virtual void
+getInput(min:int,max:int): void
+convertToLowercase(input:string): string
<<enum>>
Suit
Hearts
Clubs
Spades
Diamonds
Red
Blue
Green
Yellow
Wild
<<enum>>
Rank
Ace
Two
Three
Four
Five
Six
Seven
Eight
Nine
Ten
Jack
Queen
King
None
Zero
One
DrawTwo
DrawFour
Skip
Reverse
Deck
-deckSize: int
-cards: vector<Card*>
+insertCard(C:Card*): void
+moveCard(receiver:Deck*,C:Card*): void
+draw(): Card*
+shuffle(): void
+isEmpty(): bool
+operator<<(out:ostream&,d:Deck&): friend ostream&
+getSize const (): int
+findValid(topCard:Card*): vector<Card*>
+returnTop(): Card*
Card
-cardRank: Rank
-cardSuit: Suit
+Rank(): enum
+Suit(): enum
+Card(cardRank:Rank,cardSuit:Suit)
+getRank(): Rank
+getSuit(): Suit
+operator==(card:const Card&): bool
+operator<<(out:ostream&,card:Card&): friend ostream&

Classes:

Card: Represents a card in the game, makes up a deck of cards

- Rank - the number or value of the card
- Suit - the suit or colour of the card
- getRank - returns the number or value on the card
- getSuit - returns the suit or colour of the card
- operator== - overrides the comparison operator to compare cards
- operator<< - overrides the << operator in order to output the contents of the card to the screen

Deck: Represents a collection of cards, can be used as a hand, a discard pile, or a draw pile

- insertCard - inserts a card into the deck, mostly used during initialization
- moveCard - takes in the pointer for the deck that the card is moving to, as well as a pointer for the card that is being moved. Removes the card from the deck and adds it to the other deck
- draw - returns the pointer for the first card on the deck, but doesn't remove the card
- shuffle - rearranges the card vector randomly
- isEmpty - checks to see if there are any cards in the cards vector, returns true if there are none
- operator<< - overrides the output operator in order to output the contents of the deck to the screen
- getSize - returns the size of the cards vector
- findValid - returns a vector of cards that the player can play on their turn
- returnTop - returns the last card in the deck

Game: Represents a game, abstract

- getInput - gets the player's input
- convertToLowercase - converts a string to lowercase

Uno:

- startGame - Runs the logic for the game Uno, calling the necessary functions to run player turns, etc. (actual code is delegated to the called methods)
- initDeck - sets up the decks to play the game
- playerTurn - Runs the human player's turn, getting input for what action to take, and calling the necessary logic to complete that task.
- aiTurn - Runs the AI Player's turn, outputs the events that happened on that turn
- help - displays the help function for the game, explaining the rules, how to play, etc.
- endGame - Ends the game and declares the winner to the player.
- reset - Resets the game to its initial state

- changeColour - changes the colour of the discard pile to the specified one (used for wild cards)
- aiChangeColour - changes the colour of the discard pile to the specified one (but used for the ai)
- drawCards - used when a draw two or draw four card is played, adds the specified number of cards to the specified player's hand
- reverseOrder - reverses the turn order (doesn't do anything because there's only two players)
- skip - skips the AI player's turn
- aiSkip - skips the human player's turn
- refillDraw - refills the draw pile when it is depleted

CrazyEights:

- startGame - Runs the logic for the game crazy eights, calling the necessary functions to run player turns, etc. (actual code is delegated to the called methods)
- initDeck - sets up the decks to play the game
- playerTurn - Runs the human player's turn, getting input for what action to take, and calling the necessary logic to complete that task.
- aiTurn - Runs the AI Player's turn, outputs the events that happened on that turn
- help - displays the help function for the game, explaining the rules, how to play, etc.
- endGame - Ends the game and declares the winner to the player.
- reset - Resets the game to its initial state
- changeSuit - changes the suit of the discard pile to the specified one (used for wild cards)
- aiChangeSuit - changes the suit of the discard pile to the specified one (but used for the ai)
- refillDraw - refills the draw pile when it is depleted

Menu: Provides an interface for the player to choose a game

- chooseGame - gets the player's input for which game they want to play
- createGame -takes in a Game pointer and starts the game