UNO Game

Day 1:

1. Initialize the deck with 4 colors and with 0-9 number, each number has two cards
2. Add special cards including “draw two”, “skip”, “wild” and “wild draw four”, which two of “draw two” and “skip”, four of “wild” and “wild draw four”
3. Shuffle the deck manually

Day 2:

1. Deal cards to the player and AI, each one gets 7 cards randomly
2. Make sure the initial card is not a special card
3. Drew cards until a valid starting card was found, reshuffling invalid cards back into the deck
4. Start programming the game loop, showing the current card and what cards does the player have.
5. Tested deck generation and dealing functionality

Day 3:

1. Added functionality for the player to:

* View their hand
* Play a valid card based on the current card in play
* Draw a card from the deck if no valid card is playable

1. Implemented special card effects for:

* **Wild cards**: Player chooses a new color
* **Wild Draw Four**: Opponent draws 4 cards if valid
* **Draw Two** and **Skip**: AI skips its turn or draws cards

1. Ensured game transitions correctly between player and AI turns

Day 4:

1. Programmed the AI to:

* Search its hand for a playable card
* Draw a card if no valid card is found
* Play special cards (with appropriate effects, like making the player draw or skipping turns)
* Randomly choose a color when playing a Wild card

1. Ensured AI logic mirrored the player's gameplay mechanics

Day 5:

1. Added logic to end the game when either the player or AI runs out of cards
2. Calculated scores based on the opponent’s remaining hand:

* Special cards: 20 points
* Wild cards: 50 points

Day 6:

1. Add replay functionality:

* Prompted the player to play another round or end the game
* Reset the deck, hands, and game state if replaying

1. Added a final score display upon exiting the game
2. Conducted extensive testing to ensure smooth gameplay across multiple rounds

Day 7:

1. Handling of invalid inputs from the player
2. Ensured proper effects for all special cards
3. Optimized the code for readability and performance:

* Refactored repetitive logic into utility methods (canPlay, canPlayDrawFour, calculateScore) This method is an older friend who taught me because it was the better way to check if the card can play or not, and I also learned and understood how it worked

The rest of days:

1. Fix bugs and test the program