GameOf21 Project

James, Matthew, Colin, & Stephanie

ALGORITHM:

- Declare global variables
    - Array to hold all cards in the deck of cards
    - Array for all used cards in a deck of cards
    - Random number generator
    - Scanner

Main Method

- Prints the instructions using method
- Loop which runs until the round is over
    - Int for sum of players cards
    - Int for sum of computers cards
    - Int for number of cards played
    - Int for number of aces the player has
    - Int for number of aces the computer has
    - Boolean to check whether the round is over or not
    - Use method set_up_deck to set up the deck (cards from 1-13, 4 times)
    - Deal the cards to the computer using get_card method
    - Check if the computer has any aces
        - Add one to number of aces if they do
    - Update the deck by using the method update_deck (remove the two dealt cards so they can’t be dealt again
    - Output to the user the second card the computer is dealt
    - If computer was dealt an ace and a 10, tell user that the computer won and set round to be over
    - Add 2 to the total number of cards played
    - Deal the cards to the player using get_card method
    - Check if the player has any aces
        - Add one to number of aces if they do
    - Update the deck by using the method update_deck (remove the two dealt cards so they can't be dealt again)
    - Output to the player their 2 cards
    - If player was dealt an ace and a 10, tell user they won and set round to be over
    - Calculate the sum of the total of the players cards
    - Output to the player their sum
    - Add the second card dealt to the player to the array for cards used (for computer AI)
    - Add 2 to the number of cards played

- Checks if the computer got 21 on their first 2 cards
    - If they did, then they win the round
- Checks if the player got 21 on their first 2 cards
    - If they did, and the computer didn't, they win the round
- If the round isn't over off of the first two card deals
    - Ask the player if they want another card or not
    - Check for valid input (if input is invalid, prompt user to re-enter response)
    - If the user says no, move on
    - If the user says yes
        - Deal another card to the player using method get_card
        - Remove the dealt card from the original card array
        - If the player is dealt an ace
            - Add one to the total number of aces
        - Output to the user the new card they were dealt
        - Add the value of the new card to the sum of the players cards
    - Ask the user what they want to do with their aces
    - Convert the first two dealt cards the computer received and add them together to find the total sum of the computer cards
    - Calls the method computer_chance to see if the computer has a good chance to draw a good card or not
    - If the method returns true
        - Deal a card to the computer using the method get_card
        - If the computer is dealt an ace
            - Add one to the total number of aces
        - Removes the dealt card from the original card aray
        - Adds the new dealt card to the sum of the computers cards
        - Outputs to the player the dealt card of the computer
        - Adds 1 to the total number of cards played
    - If the computer reaches a sum of over 21 at any point, end the round
    - If the method returns false, end the round
- If the computer has any aces use method get_new_computer_number to maximize the use of the ace(s)
- Output to the player the total of the players cards
- Output to the player the total of the computers cards
- Output to the player the winner of the round and determine the winner by using the method declare_winner
- Ask the user if they want to play again
- Check for valid input (if input is invalid, prompt user to re-enter response)
- If the user enters n, end the program

```
        - If the user enters y, restart the round
```

Boolean method computer_chance

```
- If the computer sum is over 21, return false immediately
- Int for number of valid cards
- Int which stores the difference between 21 and the sum the computer has
- Loop for which runs for the number of cards already used
    - If the index of the array used cards isn't equal to 0, and the card falls into the
      difference between 21 and the sum
        - Add one to9 the number of valid cards
- If (number of valid cards / (52 - number of cards the player has) * 100 >= )
    - Return true
- Otherwise return false
```

Int method get_new_computer_number

```
- Loop that runs for the number of aces in the game
    - If the computer sum is less than 21-10, and the computer has an ace
        - Switch the ace from a 1 to an 11 and add 10 to the computer sum
    - Otherwise break
- Return the new sum of the computer cards
```

Int method get_card

```
- Randomly generate a number from 0-52 (number of cards in a deck)
- If the rng returns anything but a 0, return the number stored at the index in the deck
- Update the size of the deck of cards
```

Int method convert

```
- If the card is less than 10, return that card
- If the card is greater than 10 (royal card) return 10
```

Int method get_new_player_number

```
- Loop for the number of aces the player has
    - Output to the user the sum of their cards
    - Ask the user what they want to do with their ace
    - Check for valid input (if input is invalid, prompt user to re-enter response)
    - If the user chooses 11, the ace turns into 11 and add 10 to the sum of the players
      cards
    - Otherwise the ace stays at 1
```

```
String method get_card
    -   If the card rng returns 1, return ace
    -   If the rng returns a number from 2-10, return that number
    -   If the rng returns 11, return jack
    -   If the rng returns 12, return queen
    -   If the rng returns 13, return king

String method declare_winner
    -   If the player sum and the computer sum are both over 21
            -   Output the computer won
    -   If the player sum is over 21
            -   Output the computer won
    -   If the computer sum is over 21
            -   Output the player won
    -   If the computer sum is less than the player sum
            -   Output the player won
    -   If the computer sum is greater than the player sum
            -   Output the computer won
    -   In any other situation output the computer won

String method print_number_format
    -   If the number is 1, return first
    -   If the number is 2, return second
    -   If the number is 3, return third
    -   If the number is 4, return fourth

Void method print_instructions
    -   Output the instructions of the game

Void method update_deck
    -   Switch the dealt card to a 0 in array

Void method update_cards_used
    -   Switch the dealt card to a 0 in used cards array

Void method set_up_deck
    -   Loop that runs 4 times
            -   Loop that runs 13 times
```

- Add the values to the array cards
- Add the values to the array cards used

TEST CASES

Valid/Invalid Input

User inputs y or Y when asked if they want another card (*valid input*)	Program randomly chooses card from deck of cards array and outputs to user what card they have been dealt (expected output✓)
User inputs n or N when asking if they want another card (*valid input*)	Program goes to computer's turn of game; outputs what card computer was dealt if computer wanted another card or tells user if the computer did not want any more cards (expected output✓)
User inputs something other than y or n when asked if they want another card (*invalid input*)	A message that invalid input has been entered is printed to screen (to user) and prompts user to re-enter their response (expected output✓)
User inputs 11 or 1 when asked what value they want their ace to be (*valid input*)	Program asks user for what value they would like their next ace to be if they have any more or outputs updated sum of cards to screen (expected output✓)
User inputs something other than 11 or 1 when asked what value they want their ace to be (*invalid input)*	A message that invalid input has been entered is printed to screen (to user) and prompts user to re-enter their response (expected output✓)
User inputs y or Y when asked if they want to play again (*valid input*)	Program restarts game from beginning - outputs second card computer was dealt (expected output✓)
User inputs n or N when asked if they want to play again (*valid input*)	Message thanking user for playing the game then ends (stops executing) (expected output✓)
User inputs something other than y or n when asked if they want to play again (*invalid input*)	A message that invalid input has been entered is printed to screen (to user) and prompts user to re-enter their response (expected output✓)

Different Game Scenarios

Computer is dealt an ace and a 10	Program tells user the computer has won right away and asks user if they want to play again (prompts for input) (expected output✓)
Player is dealt an ace and a 10	Program tells user they have won right away and asks the user if they want to play again (expected output✓)
User gets an ace (or multiple aces)	Program initially assumes the value of the ace is 1 when calculating the total sum of cards of the player. Then once player does not want any more cards dealt, program asks them what value they want each other their aces to be (depending on how many they got) then updates the sum of their cards accordingly (expected output✓)
Computer gets an ace (or multiple aces)	Program initially sets sum of computer's cards based on values of aces being 1 and computer bases it chances (finds its percentage of probability of getting a card from the deck that will not bring its sum over 21) on this sum. Once it is done taking its cards, it checks if the ace will bring its sum over 21 if it is 11 and chooses accordingly. The final sum of the computer's cards is outputted to the screen. (expected output✓)
If player's sum goes over 21	Program tells user their sum went over 21, outputs the current sum of the player and computer then tells user that the computer won (expected output✓)
If computer's sum goes over 21	Program tells user the computer's sum went over 21, outputs the current sum of the player and computer then tells user that the player won (expected output✓)
If both computer and player get 21 (not from the first two cards dealt)	Program outputs both the player and computer's sum as 21 and tells user that the computer won (in game, if tied, computer automatically wins) (expected output✓)
If computer and player get same final sum	Program outputs both the player and computer's sum and tells user that the computer won (in game, if tied, computer automatically wins) (expected output✓)
If player's sum is greater than and computer's sum (but less than 21)	Program outputs both the player and computer's sum and tells user that they won (in game, if tied, computer automatically wins) (expected output✓)
If computer's sum is greater than and player's sum (but less than 21)	Program outputs both the player and computer's sum and tells user that the computer won (in game, if tied, computer automatically wins) (expected output✓)

https://codeshare.io/adNg0y // Link with code for game (might not be working)