Points possible: 100

URL to GitHub Repository:

https://github.com/MiloGonz1990/Promineo_Tech_Bootcamp/tree/main/WarCardGame

URL to Public Link of your Video: https://www.youtube.com/watch?v=jcQTrNtQ-1o

———————————————————————————————————————

Instructions:

1. Follow the **Coding Steps** below to complete this assignment.

1. In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
1. Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo.
1. Create a video showcasing your work:
   1. In this video: record and present your project verbally while showing the results of the working project.
   2. Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
   3. Your video should be a maximum of 5 minutes.
   4. Upload your video with a public link.
   5. Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.

   2. In addition, please include the following in your Coding Assignment Document:

2. The URL for this week's GitHub repository.
3. The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

1. Push the .pdf to the GitHub repo for this week.
2. Upload the .pdf to the LMS in your Coding Assignment Submission.

———————————————————————————————————————

Coding Steps — Java Final Project:

For the final project you will be creating an automated version of the classic card game *WAR*.

1. Create the following classes:
   a. Card

        i. Fields
            1. **value** (contains a value from 2-14 representing cards 2-Ace)
            2. **name** (e.g. Ace of Diamonds, or Two of Hearts)
        ii. Methods
            1. Getters and Setters
            2. **describe** (prints out information about a card)
    b. Deck
        i. Fields
            1. **cards** (List of Card)
        ii. Methods
            1. **shuffle** (randomizes the order of the cards)
            2. **draw** (removes and returns the top card of the Cards field)
            3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
    c. Player
        i. Fields
            1. **hand** (List of Card)
            2. **score** (set to 0 in the constructor)
            3. name
        ii. Methods
            1. **describe** (prints out information about the player and calls the describe method for each card in the Hand List)
            2. **flip** (removes and returns the top card of the Hand)
            3. **draw** (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
            4. **incrementScore** (adds 1 to the Player's score field)

2. Create a class called App with a main method.
    a. Instantiate a Deck and two Players, call the shuffle method on the deck.
    b. Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
    c. Using a traditional for loop, iterate 26 times and call the flip method for each player.
    d. Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the player whose card has the higher value.
    e. After the loop, compare the final score from each player.
    f. Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending on which score is higher or if they are both the same.

3. Tips: Printing out information throughout the game adds value including easier debugging as you progress and a better user experience.
    a. Using the Card describe() method when each card is flipped illustrates the game play.
    b. Printing the winner of each turn adds interest.
    c. Printing the updated score after each turn shows game progression.
    d. At the end of the game: print the final score of each player and the winner's name or "Draw" if the result is a tie.