

URL to GitHub Repository:

<https://github.com/ProjectGrantwood/Week06FinalProject>

URL to Public Link of your Video: <https://youtu.be/ZL-ZPRUllbU>

---

**Instructions:**

1. Follow the **Coding Steps** below to complete this assignment.
    - 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.
    - Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo.
    - Create a video showcasing your work:
      - In this video: record and present your project verbally while showing the results of the working project.
      - 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.
      - Your video should be a maximum of 5 minutes.
      - Upload your video with a public link.
      - 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:
    - The URL for this week's GitHub repository.
    - The URL of the public link of your video.
  3. Save the Coding Assignment Document as a .pdf and do the following:
    - Push the .pdf to the GitHub repo for this week.
    - 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*.

Create the following classes:

Card

Fields

**value** (contains a value from 2-14 representing cards 2-Ace)

**name** (e.g. Ace of Diamonds, or Two of Hearts)

Methods

Getters and Setters

**describe** (prints out information about a card)

Deck

Fields

**cards** (List of Card)

Methods

**shuffle** (randomizes the order of the cards)

**draw** (removes and returns the top card of the Cards field)

In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.

Player

Fields

**hand** (List of Card)

**score** (set to 0 in the constructor)

**name**

Methods

**describe** (prints out information about the player and calls the describe method for each card in the Hand List)

**flip** (removes and returns the top card of the Hand)

**draw** (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)

**incrementScore** (adds 1 to the Player's score field)

Create a class called App with a main method.

- Instantiate a Deck and two Players, call the shuffle method on the deck.
- Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
- Using a traditional for loop, iterate 26 times and call the flip method for each player.
- 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.
- After the loop, compare the final score from each player.
- 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.

Tips: Printing out information throughout the game adds value including easier debugging as you progress and a better user experience.

- Using the Card describe() method when each card is flipped illustrates the game play.
- Printing the winner of each turn adds interest.
- Printing the updated score after each turn shows game progression.
- 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.