https://github.com/lixy1979/java_week_06

https://youtu.be/CxJqPpfA6a0

```
package card_game_WAR;

import java.util.List;

public class Card {

	/*
	* 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)
	*/

	int value;
	String name;
	Card (String name, int value){
		this.value = value;
		this.name = name;
	}

	public int getValue() {
		return value;
	}
	public void setValue(int value) {
		this.value = value;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}

	public void describe() {
		System.out.println(this.name + " --- " + this.value);
	}
}

package card_game_WAR;
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Deck {

	/* 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.
	 */

	List<Card> cards = new ArrayList<Card>();
	Deck(){
	  String[] names = {"Clubs", "Diamonds", "Hearts", "Spades"};
	  String[] numbers = {"Two", "Three", "Four", "Five", "Six", "Seven",
	                      "Eight", "Nine", "Ten", "Jack", "Queen", "King",
	                      "Ace"};
	  for (String name : names) {
	        int i = 2;
	        for ( String number : numbers) {
	              String tmpName = number + " of " + name;
	              int value = i++;
	              Card card = new Card (tmpName, value);
	              this.cards.add(card);
	        }
	   }
	}
	 public List<Card> getCards() {
	       return cards;
	 }
	 public void setCards(List<Card> cards) {
	       this.cards = cards;
	 }
	 public void describe() {
	       for(Card card : this.cards) {
	             card.describe();
	       }
	 }

	 public void shuffle() {
	       Collections.shuffle(cards);
	 }
```

```
	public Card draw() {
		Card card = this.cards.remove(0);
		return card;
	}
```

```
package card_game_WAR;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Player {

	/*
	 * 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)
	 */

	List<Card> hand = new ArrayList<Card>();
	Deck deck = new Deck();
	int score;
	String name;

	public List<Card> getHand() {
		return hand;
	}

	public void setHand(List<Card> hand) {
		this.hand = hand;
	}

	public int getScore() {
		return score;
	}

	public void setScore(int score) {
```

```
		this.score = score;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void describe() {
		System.out.println(this.name);
		System.out.println("--------");
		for(Card card : this.hand) {
			card.describe();
		}
	}

	public Card flip() {
		return hand.remove(0);

	}
	public void  draw(Deck deck) {
		getHand().add(deck.cards.remove(0));
	}
	public void incrementScore() {
		this.score += 1;
	}

}


package card_game_WAR;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class App{
	public static void main (String[]args) {




// 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)
        System.out.println("\n Question 1 a: Card class");
            Card card = new Card("Two of Hearts" , 2);
        card.describe();
// ii.Methods
// 1.Getters and Setters
// 2.describe (prints out information about a card)

// b.Deck

// i.Fields
// 1.cards (List of Card)
    System.out.println("\n Question 1 b: Card class");
    System.out.println("\n Question 1 b deck: Card class");
    Deck deck = new Deck();
    deck.describe();

//  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.
    System.out.println("\n Question 1 b shuffle: Card class");
    deck.shuffle();
    deck.describe();
    System.out.println("\n Question 1 b draw: Card class");
    Card drawncard = deck.draw();
    drawncard.describe();

//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.
   System.out.println("\n Question 2 : App class");
   Player player1 = new Player();
   player1.setName("Tom");
   Player player2 = new Player();
   player2.setName("Jack");
   System.out.println("Shuffle Card---");
   deck.shuffle();
   System.out.println("Start the game---");
   int size = deck.cards.size();
   for (int i = 0; i < size; i++) {
        if(i %2 == 0) {
              Card card1 = deck.draw();
            player1.hand.add(card1);
        }
        else {
              Card card2 = deck.draw();
            player2.hand.add(card2);
        }
   }
   for (int i = 0; i < size / 2; i++) {
         Card card1 = player1.flip();
         System.out.println(player1.getName() + " : ");
         card1.describe();
         Card card2 = player2.flip();
         System.out.println(player2.getName() + " : ");
         card2.describe();
         System.out.println("__________");
         if(card1.getValue() > card2.getValue()) {
               player1.incrementScore();

         }else if(card2.getValue() > card1.getValue()){
               player2.incrementScore();
         }
         System.out.println(player1.getName() + " score: " +
player1.getScore());
         System.out.println(player2.getName() + " score: " +
player2.getScore());
         System.out.println("__________");
   }
   System.out.println("The final score---");
   System.out.println(player1.getName() + "' score is " +
Integer.toString(player1.getScore()));
   System.out.println(player2.getName() + "' score is " +
Integer.toString(player2.getScore()));

    if (player1.getScore() > player2.getScore()) {
```

```
            System.out.println(player1.getName() + " win the game!!!");
      } else if (player1.getScore() < player2.getScore()) {
            System.out.println(player2.getName()+ " win the game!!!");
      }else {
            System.out.println("Draw");
      }
//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.
      }
}
```