```swift
import UIKit
//Sets up a class for the buttons used in the game
class customButton: UIButton {
    override func layoutSubviews(){
        super.layoutSubviews()
        layer.cornerRadius = 15
        layer.masksToBounds = true
        layer.opacity = 0.7
    }
}
class ViewController: UIViewController {

    //Initial variables
    var deck: Deck = Deck()
    var dealerCount = 0
    var playerCount = 0
    var turn = true
    var dealerTotal: Int = 0
    var playerTotal: Int = 0
    var playersHand: [Int] = []
    var dealersHand: [Int] = []
    var newDeck: [Card] = []
    var gameStage = 0

    //Outlets for things in the app
    @IBOutlet weak var playerScoreLabel: UILabel!
    @IBOutlet weak var dealerScoreLabel: UILabel!
    @IBOutlet weak var infoLabel: UILabel!
    @IBOutlet weak var cardImage1: UIImageView!
    @IBOutlet weak var cardImage2: UIImageView!

    //Code for when the player hits
    @IBAction func hitButtonPressed(_ sender: Any) {
        createCard()
    }

    //Code for when the player stands (checks winner)
    @IBAction func standButtonPressed(_ sender: Any) {

        dealerTurn()

        if playerTotal > dealerTotal && playerTotal <= 21 && dealerTotal <= 21{
            print("You win!")
            infoLabel.text = "You had a hand of \(playerTotal) and the dealer had a hand of \(dealerTotal)! You won!"
        }else if playerTotal < dealerTotal && dealerTotal <= 21{
            print("You lose!")
            infoLabel.text = "You had a hand of \(playerTotal) and the dealer had a hand of \(dealerTotal)! You lost!"
        }else if playerTotal == dealerTotal{
            print ("You tied!")
            infoLabel.text = "You had a hand of \(playerTotal) and the dealer had a hand of \(dealerTotal)! You tied!"
        }else{
            print ("Something happened")
        }
        dealerScoreLabel.text = "Dealer: \(dealerTotal)"
    }

    //Restarts the game after a delay of 3 seconds
    @IBAction func restartButtonPressed(_ sender: Any) {
        infoLabel.text = "Restarting..."
        DispatchQueue.main.asyncAfter(deadline: .now() + 3){
```

```
        self.dealerTotal = 0
        self.playerTotal = 0
        self.dealerCount = 0
        self.playerCount = 0
        self.playersHand = []
        self.dealersHand = []
        self.gameStage = 0
        self.startGame()
    }
}

//Generates a card for the player
func createCard(){
    if playerCount < 7{
        playerTurn()
    }else{
        print("No more cards")
        }
    }

//Checks if anybody has busted, and displays it in the info label if so
func infoUpdate(){
    if dealerTotal > 21{
        infoLabel.text = "The dealer busted, you win!"
    }
    if playerTotal > 21{
        infoLabel.text = "You busted!"
    }else if playerTotal == 21{
        infoLabel.text = "Congratulations! You won with a hand of exactly 21!"
    }
}

//Code for the dealer's turn, when the player has finished drawing cards
func dealerTurn(){

    if gameStage == 1{
        let randomIndex = Int.random(in: 0..<newDeck.count)
        let card = newDeck[randomIndex].image
        while dealerTotal < 17{
                self.cardImage1.image = UIImage(named: card)
                self.dealersHand.append(Int(card.suffix(2))!)
                print((self.dealersHand))
                self.dealerTotal = self.dealersHand.reduce(0, +)
                print(self.dealerTotal)
                self.infoUpdate()
                self.dealerCount += 1
        }
        if dealerTotal >= 17{
            cardImage1.image = UIImage(named: card)
        }
    }
    if gameStage == 0{
        let randomIndex = Int.random(in: 0..<newDeck.count)
        let card = newDeck[randomIndex].image
            self.cardImage1.image = UIImage(named: card)
            self.dealersHand.append(Int(card.suffix(2))!)
            print((self.dealersHand))
            self.dealerTotal = self.dealersHand.reduce(0, +)
            print(self.dealerTotal)
            self.infoUpdate()
            self.dealerCount += 1
```

```
            self.turn = false

        }
    }

    //Code for the player's turn
    func playerTurn(){
        let randomIndex = Int.random(in: 0..<newDeck.count)
        let card = newDeck[randomIndex].image
        cardImage2.image = UIImage(named: card)
        playersHand.append(Int(card.suffix(2))!)
        print((playersHand))
        playerTotal = playersHand.reduce(0, +)
        playerScoreLabel.text = "Player: \(playerTotal)"
        print (playerTotal)
        playerCount += 1
        infoUpdate()
    }

    //Code for starting a new round
    func startGame(){
        newDeck = deck.createDeck()
        newDeck.shuffle()
        playerTurn()
        playerTurn()
        dealerTurn()
        dealerTurn()
        gameStage = 1
        print (gameStage)
        cardImage1.image  = UIImage.named( "card_back" )
        infoUpdate()
        dealerScoreLabel.text = "Dealer: ?"
        infoLabel.text = "It is your turn; you can choose to hit or stay."
    }

    //Code that runs when the game loads for the first time
    override func viewDidLoad() {
      startGame()
        cardImage2.alpha = 0.8
        cardImage1.alpha = 0.8
    }
}
```

```swift
import Foundation
//Sets up the foundations for how to make a card
class Card{
    var suit: String
    var rank: String
    var value: Int
    var image: String

    init(suit: String, rank: String, value: Int) {
        self.suit = suit
        self.rank = rank
        self.value = value
        self.image = value < 10 ? suit + "0" + String(value) : suit + String(value)
    }
}
//how to make a deck of cards, using the card class set up above
class Deck{
    var cards: [Card]

    init(){
        self.cards = Array<Card>()
    }
    func createDeck() -> [Card]{
        let suits = ["Hearts", "Diamonds", "Clubs", "Spades"]
        let ranks = ["Ace": 1, "Two": 2, "Three": 3, "Four": 4, "Five": 5, "Six": 6, "Seven": 7, "Eight": 8, "Nine": 9, "King":
10, "Queen": 10, "Jack": 10]

        var blankDeck: [Card] = []

        for suit in suits{
            for rank in ranks{
                let newCard = Card(suit: suit, rank: rank.key, value: rank.value)
                blankDeck.append(newCard)
            }
        }
        return blankDeck
    }
}
```