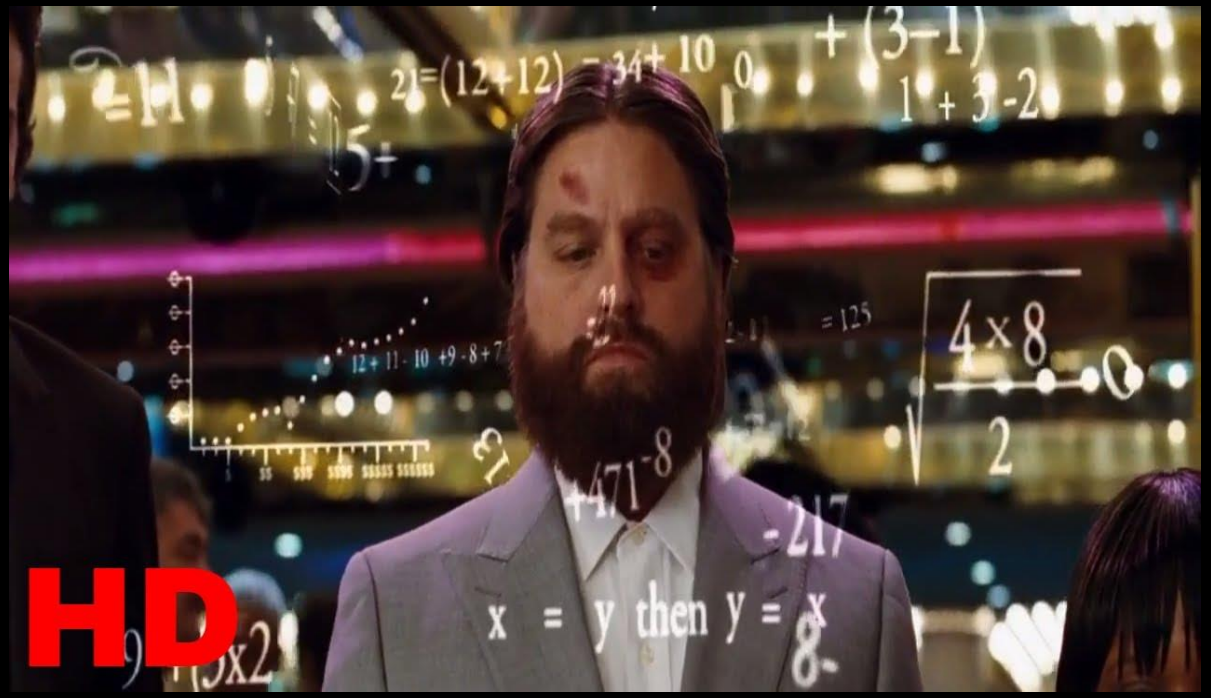
MARKUS AND THOMAS

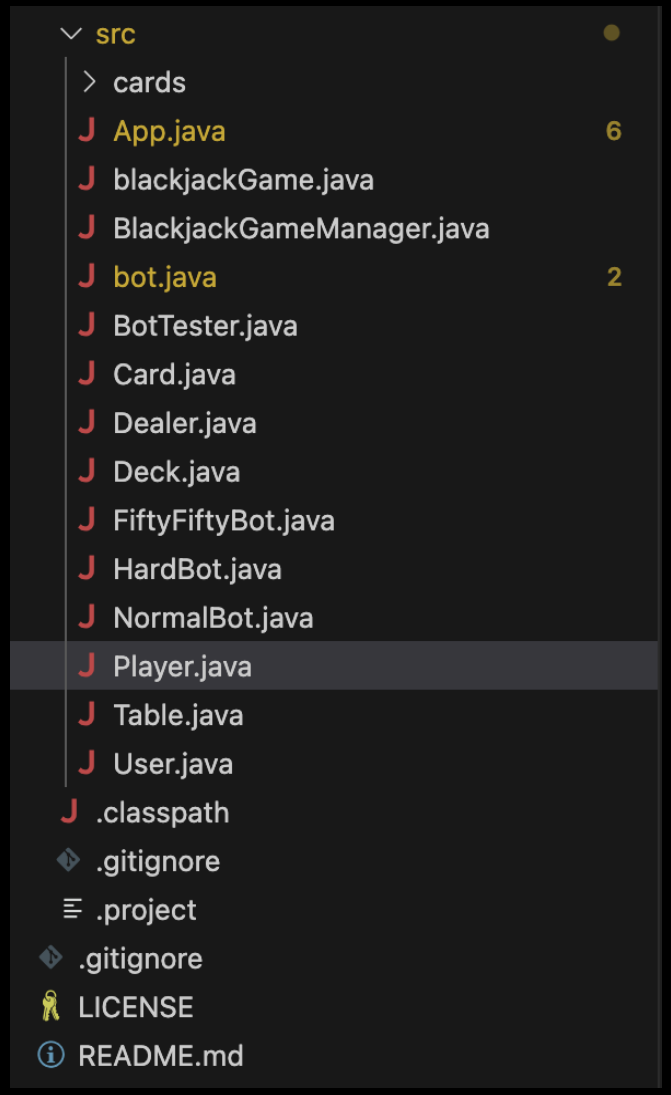
# JAVA BLACK JACK PROJECT



# BLACK JACK RULES

- Main Goal: Get to 21
- Do not go over 21
- Aces count for either 1 or 11, and faces count for 10
- You are dealt 2 cards first and you are given the option to hit or stand
- If the dealer gets a higher sum than you without busting they also win

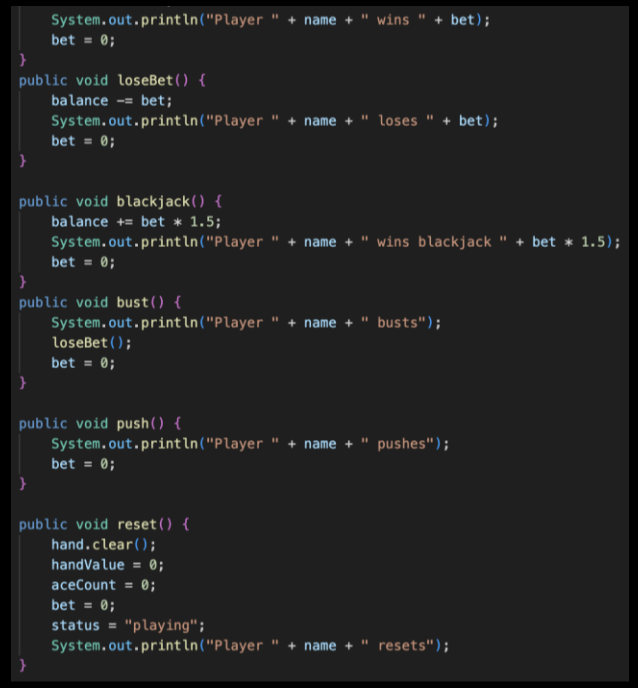
# HOW DOES THE CODE WORK?




# BLACK JACK GAME MANAGER

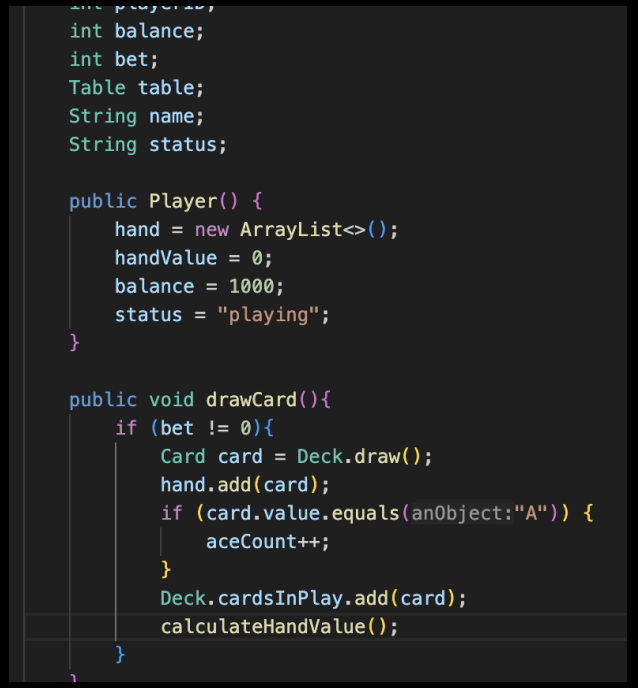
```java
public class BlackjackGameManager {

    public BlackjackGameManager() {
        blackjackGame game = new blackjackGame();
        game.initializeGame();
        Table table = new Table(game);
        game.tableToPlayers(table);
        App.window.add(table);
        App.window.setResizable(resizable:false);
        App.window.pack();
        App.window.setLocationRelativeTo(c:null);
        App.window.setVisible(b:true);
        App.window.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        Deck.newDeck();
        Deck.shuffle();
        do {
            System.out.println(x:"Round started");
            System.out.println("Deck size: " + Deck.deck.size());
            game.resetGame();
            table.repaint();
            game.getBets();
            table.repaint();
            game.dealCards();
            table.repaint();
            game.hitOrStand();
            table.repaint();
            game.checkBlackjack();
            table.repaint();
            game.dealerTurn();
            table.repaint();
            game.checkWinners();
            table.repaint();
        } while (game.playAgain());
        App.window.remove(table);
    }


}
```

# THE PLAYER CLASS



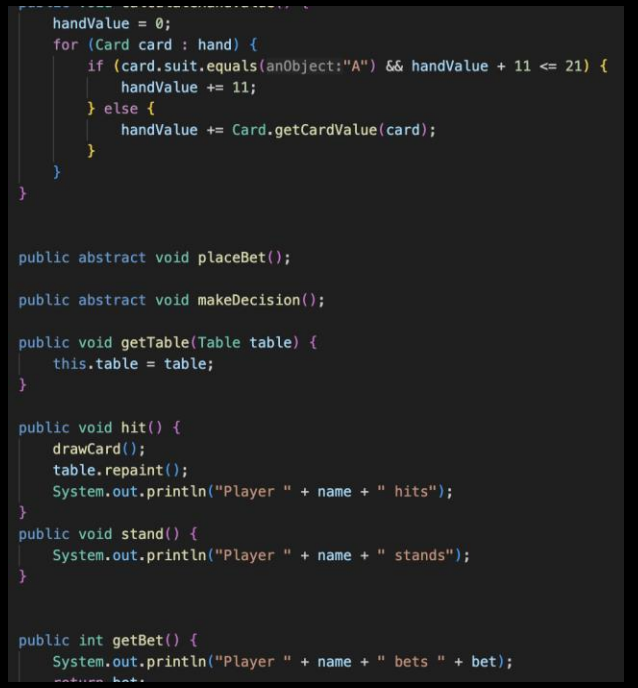
```
    System.out.println("Player " + name + " wins " + bet);
    bet = 0;
}
public void loseBet() {
    balance -= bet;
    System.out.println("Player " + name + " loses " + bet);
    bet = 0;
}

public void blackjack() {
    balance += bet * 1.5;
    System.out.println("Player " + name + " wins blackjack " + bet * 1.5);
    bet = 0;
}
public void bust() {
    System.out.println("Player " + name + " busts");
    loseBet();
    bet = 0;
}

public void push() {
    System.out.println("Player " + name + " pushes");
    bet = 0;
}

public void reset() {
    hand.clear();
    handValue = 0;
    aceCount = 0;
    bet = 0;
    status = "playing";
    System.out.println("Player " + name + " resets");
}
```

Methods



```
    int balance;
    int bet;
    Table table;
    String name;
    String status;

    public Player() {
        hand = new ArrayList<>();
        handValue = 0;
        balance = 1000;
        status = "playing";
    }

    public void drawCard(){
        if (bet != 0){
            Card card = Deck.draw();
            hand.add(card);
            if (card.value.equals(anObject:"A")) {
                aceCount++;
            }
            Deck.cardsInPlay.add(card);
            calculateHandValue();
        }
```

Variable Declarations



```
    handValue = 0;
    for (Card card : hand) {
        if (card.suit.equals(anObject:"A") && handValue + 11 <= 21) {
            handValue += 11;
        } else {
            handValue += Card.getCardValue(card);
        }
    }
}


public abstract void placeBet();

public abstract void makeDecision();

public void getTable(Table table) {
    this.table = table;
}

public void hit() {
    drawCard();
    table.repaint();
    System.out.println("Player " + name + " hits");
}
public void stand() {
    System.out.println("Player " + name + " stands");
}


public int getBet() {
    System.out.println("Player " + name + " bets " + bet);
```

Abstract Method Creation

# THE AI(S)

```java
import java.util.Random;
public class FiftyFiftyBot extends Player {
    Random random = new Random();

    public FiftyFiftyBot(){
        super();
        this.name = "FiftyFiftyBot";
        playerID = 2;
    }

    @Override
    public void placeBet(){
        bet = random.nextInt(origin:0,balance);
        status = "Bet Amount: " + bet;
    }

    @Override
    public void makeDecision(){
        calculateHandValue();
        if (handValue<21){
        int randomNumberInRange = random.nextInt(origin:0, bound:2);
        if(randomNumberInRange == 0){
            hit();
        }
        else{
            stand();
        }
}
    }

}
```

FiftyFiftyBot

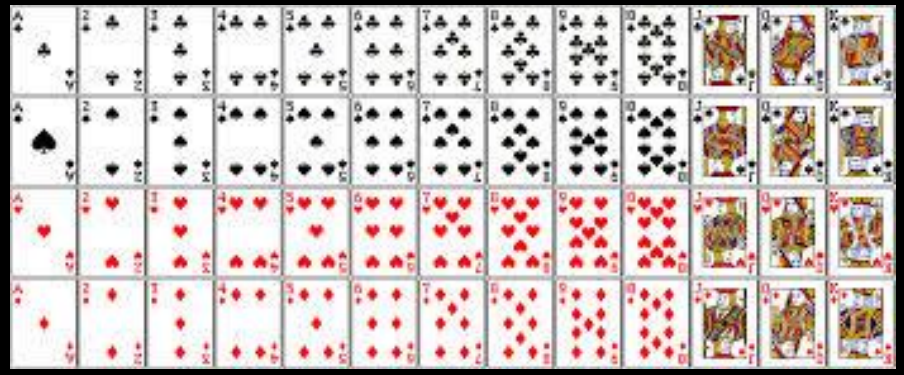
```java
public class HardBot extends Player{
    public static double calculateExpectedValue(ArrayList<Card> deck) {
        double deckValue = 0.0;
        double cardNumber = 0.0;
        for (Card card : deck) {
            if (card.suit.equals(anObject:"A") && deckValue + 11 <= 21) {
                deckValue += 11;
                cardNumber++;
            } else {
                deckValue += Card.getCardValue(card);
                cardNumber++;
            }
        }
        return deckValue/cardNumber;
    }

    public HardBot(){
        super();
        this.name = "HardBot";
        playerID = 4;
    }

    @Override
    public void placeBet(){
        bet = 3*balance/10;
        status = "Bet Amount: " + bet;
    }

    @Override
    public void makeDecision(){
        calculateHandValue();
        if (handValue<16 && HardBot.calculateExpectedValue(Deck.deck)<6.5385){
            hit();
        }
        else if (handValue<17 && HardBot.calculateExpectedValue(Deck.deck)<6){
            hit();
        }
        else if(handValue<18 && HardBot.calculateExpectedValue(Deck.deck)<5){
            hit();
        }
        else if(handValue<19 && HardBot.calculateExpectedValue(Deck.deck)<4){
            hit();}
        else if(handValue<20 && HardBot.calculateExpectedValue(Deck.deck)<3){
            hit();}
        else if(handValue<21 && HardBot.calculateExpectedValue(Deck.deck)<2){
            hit();}
```

HardBot

```java
import java.util.Random;

public class NormalBot extends Player {

    Random random = new Random();

    public NormalBot() {
        super();
        this.name = "NormalBot";
        playerID = 3;
    }

    @Override
    public void placeBet() {
        if (balance < 50) {
            bet = balance;
        } else {
            bet = 2 * balance / 10;
        }
        status = "Bet Amount: " + bet;
    }

    @Override
    public void makeDecision() {
            calculateHandValue();
        if (handValue < 21) {
            int randomNumberInRange = random.nextInt(bound:2);
            if (randomNumberInRange == 0) {
                hit();
                System.out.println(x:"NormalBot hits");
            } else {
                stand();
                System.out.println(x:"NormalBot stands");
            }
        }
    }
}
```

NormalBot

# THE DECK



```java
public class Deck {
    public static ArrayList<Card> deck;
    public static ArrayList<Card> cardsInPlay = new ArrayList<>();// Does not incluce hidden Cards

    public Deck() {

    }

    public static void newDeck() {
        deck = new ArrayList<>();
        String[] suits = {"H", "D", "C", "S"};
        String[] values = {"A","2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};
        for (String suit : suits) {
            for (String value : values) {
                deck.add(new Card(suit, value));
                deck.add(new Card(suit, value));
            }
        }
        System.out.println("New deck: " + deck + "\n" + "Deck size: " + deck.size());
    }

    public static void shuffle() {
        Random random = new Random();
        for (int i = 0; i < deck.size(); i++) {
            int r = random.nextInt(deck.size());
            Card temp = deck.get(i);
            deck.set(i, deck.get(r));
            deck.set(r, temp);
        }
        System.out.println("Shuffled Deck: " + deck);
    }

    public static Card draw() {
        if (deck.isEmpty()) {
            System.out.println(x:"Deck is empty");
            return null;
        }
        Card card = deck.get(index:0);
        deck.remove(index:0);
        System.out.println("Drew card: " + card + "\n" + "Deck size: " + deck.size());
        return card;
    }
```

# WHO WANTS TO PLAY?