Nama : Muhamad Yusuf Ardabili
NIM : 2222105132
Kelas : 2TI03

Codingan

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.*;

public class BlackJack {
    private class Card {
        String value;
        String type;

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

        public String toString() {
            return value + "-" + type;
        }

        public int getValue() {
            if ("AJQK".contains(value)) { //A J Q K
                if (value == "A") {
                    return 11;
                }
                return 10;
            }
            return Integer.parseInt(value); //2-10
        }

        public boolean isAce() {
            return value == "A";
        }

        public String getImagePath() {
            return "./cards/" + toString() + ".png";
        }
    }

    ArrayList<Card> deck;
    Random random = new Random(); //shuffle deck

    //dealer
```

```
    Card hiddenCard;
    ArrayList<Card> dealerHand;
    int dealerSum;
    int dealerAceCount;

    //player
    ArrayList<Card> playerHand;
    int playerSum;
    int playerAceCount;

    //window
    int boardWidth = 600;
    int boardHeight = boardWidth;

    int cardWidth = 110; //ratio should 1/1.4
    int cardHeight = 154;

    JFrame frame = new JFrame("Black Jack");
    JPanel gamePanel = new JPanel() {
        @Override
        public void paintComponent(Graphics g) {
            super.paintComponent(g);

            try {
                //draw hidden card
                Image hiddenCardImg = new
ImageIcon(getClass().getResource("./cards/BACK.png")).getImage();
                if (!stayButton.isEnabled()) {
                    hiddenCardImg = new
ImageIcon(getClass().getResource(hiddenCard.getImagePath())).getImage();
                }
                g.drawImage(hiddenCardImg, 20, 20, cardWidth, cardHeight,
null);

                //draw dealer's hand
                for (int i = 0; i < dealerHand.size(); i++) {
                    Card card = dealerHand.get(i);
                    Image cardImg = new
ImageIcon(getClass().getResource(card.getImagePath())).getImage();
                    g.drawImage(cardImg, cardWidth + 25 + (cardWidth + 5)*i,
20, cardWidth, cardHeight, null);
                }

                //draw player's hand
                for (int i = 0; i < playerHand.size(); i++) {
                    Card card = playerHand.get(i);
                    Image cardImg = new
ImageIcon(getClass().getResource(card.getImagePath())).getImage();
```

```
                    g.drawImage(cardImg, 20 + (cardWidth + 5)*i, 320,
cardWidth, cardHeight, null);
                }

                if (!stayButton.isEnabled()) {
                    dealerSum = reduceDealerAce();
                    playerSum = reducePlayerAce();
                    System.out.println("STAY: ");
                    System.out.println(dealerSum);
                    System.out.println(playerSum);

                    String message = "";
                    if (playerSum > 21) {
                        message = "You Lose!";
                    }
                    else if (dealerSum > 21) {
                        message = "You Win!";
                    }
                    //both you and dealer <= 21
                    else if (playerSum == dealerSum) {
                        message = "Tie!";
                    }
                    else if (playerSum > dealerSum) {
                        message = "You Win!";
                    }
                    else if (playerSum < dealerSum) {
                        message = "You Lose!";
                    }

                    g.setFont(new Font("Arial", Font.PLAIN, 30));
                    g.setColor(Color.white);
                    g.drawString(message, 220, 250);
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
    JPanel buttonPanel = new JPanel();
    JButton hitButton = new JButton("Hit");
    JButton stayButton = new JButton("Stay");

    BlackJack() {
        startGame();

        frame.setVisible(true);
        frame.setSize(boardWidth, boardHeight);
```

```
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        gamePanel.setLayout(new BorderLayout());
        gamePanel.setBackground(new Color(53, 101, 77));
        frame.add(gamePanel);

        hitButton.setFocusable(false);
        buttonPanel.add(hitButton);
        stayButton.setFocusable(false);
        buttonPanel.add(stayButton);
        frame.add(buttonPanel, BorderLayout.SOUTH);

        hitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Card card = deck.remove(deck.size()-1);
                playerSum += card.getValue();
                playerAceCount += card.isAce() ? 1 : 0;
                playerHand.add(card);
                if (reducePlayerAce() > 21) { //A + 2 + J --> 1 + 2 + J
                    hitButton.setEnabled(false);
                }
                gamePanel.repaint();
            }
        });

        stayButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                hitButton.setEnabled(false);
                stayButton.setEnabled(false);

                while (dealerSum < 17) {
                    Card card = deck.remove(deck.size()-1);
                    dealerSum += card.getValue();
                    dealerAceCount += card.isAce() ? 1 : 0;
                    dealerHand.add(card);
                }
                gamePanel.repaint();
            }
        });

        gamePanel.repaint();
    }

    public void startGame() {
        //deck
        buildDeck();
```

```
        shuffleDeck();

        //dealer
        dealerHand = new ArrayList<Card>();
        dealerSum = 0;
        dealerAceCount = 0;

        hiddenCard = deck.remove(deck.size()-1); //remove card at last index
        dealerSum += hiddenCard.getValue();
        dealerAceCount += hiddenCard.isAce() ? 1 : 0;

        Card card = deck.remove(deck.size()-1);
        dealerSum += card.getValue();
        dealerAceCount += card.isAce() ? 1 : 0;
        dealerHand.add(card);

        System.out.println("DEALER:");
        System.out.println(hiddenCard);
        System.out.println(dealerHand);
        System.out.println(dealerSum);
        System.out.println(dealerAceCount);


        //player
        playerHand = new ArrayList<Card>();
        playerSum = 0;
        playerAceCount = 0;

        for (int i = 0; i < 2; i++) {
            card = deck.remove(deck.size()-1);
            playerSum += card.getValue();
            playerAceCount += card.isAce() ? 1 : 0;
            playerHand.add(card);
        }

        System.out.println("PLAYER: ");
        System.out.println(playerHand);
        System.out.println(playerSum);
        System.out.println(playerAceCount);
    }

    public void buildDeck() {
        deck = new ArrayList<Card>();
        String[] values = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"J", "Q", "K"};
        String[] types = {"C", "D", "H", "S"};

        for (int i = 0; i < types.length; i++) {
            for (int j = 0; j < values.length; j++) {
```

```
                Card card = new Card(values[j], types[i]);
                deck.add(card);
            }
        }

        System.out.println("BUILD DECK:");
        System.out.println(deck);
    }

    public void shuffleDeck() {
        for (int i = 0; i < deck.size(); i++) {
            int j = random.nextInt(deck.size());
            Card currCard = deck.get(i);
            Card randomCard = deck.get(j);
            deck.set(i, randomCard);
            deck.set(j, currCard);
        }

        System.out.println("AFTER SHUFFLE");
        System.out.println(deck);
    }

    public int reducePlayerAce() {
        while (playerSum > 21 && playerAceCount > 0) {
            playerSum -= 10;
            playerAceCount -= 1;
        }
        return playerSum;
    }

    public int reduceDealerAce() {
        while (dealerSum > 21 && dealerAceCount > 0) {
            dealerSum -= 10;
            dealerAceCount -= 1;
        }
        return dealerSum;
    }
}
```

Penjelasan Codingan

1. Kelas Card

Kelas ini merepresentasikan sebuah kartu dalam permainan Blackjack. Setiap kartu memiliki nilai (`value`) dan tipe (`type`), seperti "A" untuk As, "2" hingga "10", "J" untuk Jack, "Q" untuk Queen, dan "K" untuk King, serta tipe kartu seperti Clubs (C), Diamonds (D), Hearts (H), dan Spades (S). Kelas ini juga memiliki beberapa metode:

- **toString()**: Mengembalikan representasi string dari kartu.

- **getValue()**: Mengembalikan nilai numerik dari kartu, di mana "A" bernilai 11 atau 1, "J", "Q", dan "K" bernilai 10, dan kartu lainnya bernilai sesuai angka.

- **isAce()**: Mengembalikan `true` jika kartu adalah As.

- **getImagePath()**: Mengembalikan path dari gambar kartu.

2. Deklarasi Variabel dan Objek

Beberapa variabel utama di kelas BlackJack:

- **deck**: ArrayList yang menyimpan tumpukan kartu.

- **random**: Objek `Random` untuk mengacak kartu.

- **hiddenCard**: Kartu dealer yang disembunyikan.

- **dealerHand**: ArrayList untuk menyimpan kartu dealer.

- **playerHand**: ArrayList untuk menyimpan kartu pemain.

- **dealerSum, dealerAceCount, playerSum, playerAceCount**: Variabel untuk menghitung total nilai dan jumlah As di tangan dealer dan pemain.

- frame, gamePanel, buttonPanel, hitButton, stayButton: Elemen GUI untuk permainan.

3. Antarmuka Grafis (GUI)

- **gamePanel**: Panel untuk menggambar kartu dan informasi permainan.

- **buttonPanel**: Panel yang berisi tombol "Hit" dan "Stay".

- **hitButton dan stayButton**: Tombol yang digunakan pemain untuk mengambil kartu tambahan atau berhenti.

4. Metode **paintComponent**

Metode ini digunakan untuk menggambar kartu pada `gamePanel`, termasuk kartu dealer dan kartu pemain. Kartu dealer yang tersembunyi digambar dengan gambar punggung kartu hingga tombol "Stay" ditekan.

5. Konstruktor **BlackJack()**

Konstruktor ini menginisialisasi permainan dengan memanggil metode `startGame()`, mengatur properti JFrame, dan menambahkan action listener untuk tombol "Hit" dan "Stay".

6. Metode **startGame()**

Metode ini memulai permainan dengan membangun tumpukan kartu (`buildDeck()`), mengacaknya (`shuffleDeck()`), dan mendistribusikan kartu awal ke dealer dan pemain.

7. Metode **buildDeck()**

Metode ini membuat tumpukan kartu dengan semua kombinasi nilai dan tipe kartu.

8. Metode **shuffleDeck()**

Metode ini mengacak tumpukan kartu menggunakan objek `Random`.

9. Metode **reducePlayerAce() dan reduceDealerAce()**

Metode ini mengurangi nilai total kartu pemain atau dealer jika mereka memiliki As dan total nilainya lebih dari 21, dengan mengubah nilai As dari 11 menjadi 1.

10. Logika Permainan

- Tombol **"Hit"**: Menambahkan kartu ke tangan pemain dan menghitung ulang nilai total. Jika nilai total pemain melebihi 21, tombol "Hit" dinonaktifkan.

- Tombol **"Stay"**: Menonaktifkan tombol "Hit" dan "Stay", kemudian dealer mengambil kartu hingga total nilainya mencapai setidaknya 17. Setelah itu, nilai total kartu pemain dan dealer dibandingkan untuk menentukan pemenang.

11. Menjalankan Aplikasi

Untuk menjalankan aplikasi ini, Anda perlu memiliki Java Development Kit (JDK) dan membuat folder berisi gambar-gambar kartu yang sesuai dengan path yang disebutkan di kode (misalnya `./cards/2-H.png` untuk kartu 2 of Hearts). Anda kemudian dapat mengkompilasi dan menjalankan program ini untuk memainkan permainan Blackjack.