Nama	: Muhamad Aidil Fahri
NIM	: 2222105127
Kelas	: 2TI03
Dosen	: Novan Zulkamain, S.T., M.Kom.
Mata Kuliah	: Pemrograman Berorientasi Objek

UAS

Codingan Game Blackjack

```
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 ("JQK".contains(value)) { // J Q K
                return 10;
            }
            if ("A".equals(value)) { // A
                return 11;
            }
            return Integer.parseInt(value); // 2-10
        }

        public boolean isAce() {
            return "A".equals(value);
        }
```

```
        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 be 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);
                }

                dealerSum = reduceDealerAce();
                playerSum = reducePlayerAce();
                System.out.println("STAY: ");
                System.out.println(dealerSum);
                System.out.println(playerSum);

                // Display player and dealer scores
                g.setFont(new Font("Arial", Font.PLAIN, 30));
                g.setColor(Color.white);
                g.drawString("Player Score: " + playerSum, 20, 300);
                if (!stayButton.isEnabled()) {
                    g.drawString("Dealer Score: " + dealerSum, 20, 200);

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

                    g.drawString(message, 220, 250);
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
};
JPanel buttonPanel = new JPanel();
JButton hitButton = new JButton("Hit");
JButton stayButton = new JButton("Stay");
JButton replayButton = new JButton("Replay");


public 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);
    replayButton.setFocusable(false);
    replayButton.setBackground(Color.RED); // Changed this line
    replayButton.setForeground(Color.WHITE); // Corrected this line
    buttonPanel.add(replayButton);
    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);
                stayButton.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();
            }
        });

        replayButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                startGame();
                hitButton.setEnabled(true);
                stayButton.setEnabled(true);
                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;
    }

    public static void main(String[] args) {
        new BlackJack();
    }
}
```

Penjelasan codingan game blackjack

1. Kelas Card :
    - `Card` adalah kelas nested (inner class) yang merepresentasikan sebuah kartu dalam permainan blackjack.
    - Memiliki atribut `value` dan `type` untuk menentukan nilai dan jenis kartu.
    - Memiliki method `getValue()` untuk mengembalikan nilai numerik dari kartu (dengan mempertimbangkan aturan nilai di Blackjack).
    - Method `isAce()` digunakan untuk mengecek apakah kartu tersebut adalah Ace.
    - `getImagePath()` mengembalikan path ke file gambar kartu, diasumsikan dalam folder "cards".

2. Variabel dan Komponen UI :
    - `ArrayList<Card> deck` menyimpan semua kartu yang akan digunakan dalam permainan.
    - Variabel `dealerHand`, `playerHand` untuk menyimpan kartu yang dimiliki dealer dan pemain saat ini.
    - `JFrame frame` adalah window utama permainan.
    - `JPanel gamePanel` digunakan untuk menggambar tampilan permainan, termasuk kartu-kartu yang ditampilkan.
    - `JPanel buttonPanel` untuk menyimpan tombol-tombol aksi (Hit, Stay, dan Replay).
    - `JButton hitButton`, `stayButton`, dan `replayButton` adalah tombol aksi untuk permainan (mengambil kartu, berhenti, dan memulai ulang permainan).

3. Metode `paintComponent(Graphics g)` :
    - Override dari JPanel untuk menggambar elemen-elemen permainan seperti kartu dealer, kartu pemain, skor, dan pesan hasil.
    - Menggunakan `Graphics` untuk menggambar gambar kartu dari path yang telah ditentukan.
    - Menampilkan skor pemain dan dealer, serta pesan hasil permainan (Menang, Kalah, Seri).

4. Metode `startGame()`:
    - Menginisialisasi permainan dengan membangun deck kartu, mengacak deck, dan membagikan kartu ke dealer dan pemain.
    - Mengatur ulang variabel-variabel terkait seperti `dealerHand`, `playerHand`, `dealerSum`, `playerSum`, dan sebagainya.

5. Metode `buildDeck()` dan `shuffleDeck()` :

- `buildDeck()` membangun deck kartu dengan menginisialisasi semua kombinasi nilai dan jenis kartu.
- `shuffleDeck()` mengacak kartu dalam deck dengan menggunakan `Random`.

6. Aksi Listener pada Tombol :
    - `hitButton.addActionListener()` untuk menambahkan kartu ke tangan pemain.
    - `stayButton.addActionListener()` untuk mengakhiri giliran pemain dan membiarkan dealer mengambil kartu.
    - `replayButton.addActionListener()` untuk memulai ulang permainan dengan mengatur ulang semua variabel dan menggambar ulang gamePanel.

7. Metode `reducePlayerAce()` dan `reduceDealerAce()` :
    - Mengelola kasus di mana kartu As (Ace) dihitung sebagai 11 atau 1 tergantung pada total skor pemain atau dealer.

8. Metode `main()` :
    - Metode utama yang membuat objek `BlackJack`, memulai permainan dengan memanggil konstruktor.

Hasil