```
public class Gioco7m {
   public static void main(String argv[]) {
      MezzoSette f = new MezzoSette();
      f.setTitle("Gioco del 7 e mezzo");
      f.setSize(600, 400);
      f.setVisible(true);
   }
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.File;

class MezzoSette extends JFrame implements ActionListener {
   private String Carte[] = new String[41];
   private int Mazzo[] = new int[41];
   private double Punti[] = new double[4];
   private int prosCarta;
   private Point x0y0 = new Point(10, 25);
   private int dxdy = 30;
   private int Giocatore = 0;
   private boolean terminato = false;
   private String[] nomiGiocatori = {"Mazziere", "Giocatore 1", "Giocatore 2", "Giocatore 3"};
   private String nomeGiocatoreCorrente;

   private JLabel Punteggio[] = new JLabel[4];
   private JLabel imgCarte[] = new JLabel[44];
   private ButtonGroup AC = new ButtonGroup();
   private JButton Carta[] = new JButton[4];
   private JButton Sto[] = new JButton[4];
   private JButton restartButton;
   private JButton quitButton;
   private JLayeredPane TavoloVerde;

   public MezzoSette() {
      String[] Semi = {"Coppe", "Denari", "Spade", "Bastoni"};

      for (int i = 0; i < 4; i++) {
         for (int j = 1; j <= 10; j++) {
            Carte[i * 10 + j] = Semi[i] + j;
```

```
    }
  }

  for (int i = 0; i < 40; i++) Mazzo[i] = i + 1;
  Mescola();

  for (int i = 0; i < 44; i++) imgCarte[i] = new JLabel();

  prosCarta = 0;
  TavoloVerde = new JLayeredPane();
  TavoloVerde.setPreferredSize(new Dimension(600, 400));

  for (int i = 0; i < 4; i++) {
    Punteggio[i] = new JLabel("0.0 (" + nomiGiocatori[i] + ")");
    TavoloVerde.add(Punteggio[i], 1);
    Punteggio[i].setBounds(5 + i * 150, 250, 105, 25);

    Carta[i] = new JButton("Carta");
    AC.add(Carta[i]);
    Font font = new Font("Arial", Font.PLAIN, 13); // Impostiamo un carattere più piccolo (dimensione
10)
    Carta[i].setFont(font); // Applichiamo il nuovo font al pulsante
    Carta[i].setBounds(5 + i * 150, 0, 70, 25); // Manteniamo le dimensioni del pulsante
    TavoloVerde.add(Carta[i], 1);
    Carta[i].setActionCommand("Carta" + i);
    Carta[i].addActionListener(this);

    Sto[i] = new JButton("Sto");
    AC.add(Sto[i]);
    Sto[i].setBounds(65 + i * 150, 0, 55, 25);
    TavoloVerde.add(Sto[i], 1);
    Sto[i].setActionCommand("Sto" + i);
```

```
   Sto[i].addActionListener(this);
}

// Adding the new buttons for restarting and quitting
restartButton = new JButton("Rinizia");
restartButton.setBounds(5, 300, 100, 25);
restartButton.setActionCommand("Rinizia");
restartButton.addActionListener(this);
TavoloVerde.add(restartButton, 1);

quitButton = new JButton("Termina");
quitButton.setBounds(120, 300, 100, 25);
quitButton.setActionCommand("Termina");
quitButton.addActionListener(this);
TavoloVerde.add(quitButton, 1);

// Stampa del percorso di lavoro corrente per il debug
System.out.println("Current Working Directory: " + System.getProperty("user.dir"));

// Carica immagini delle carte
caricaImmaginiCarte();

TavoloVerde.add(imgCarte[0], new Integer(0));
TavoloVerde.add(imgCarte[41], new Integer(0));
TavoloVerde.add(imgCarte[42], new Integer(0));
TavoloVerde.add(imgCarte[43], new Integer(0));

imgCarte[0].setBounds(x0y0.x, x0y0.y, 55, 101);
imgCarte[41].setBounds(x0y0.x + 150, x0y0.y, 55, 101);
imgCarte[42].setBounds(x0y0.x + 300, x0y0.y, 55, 101);
imgCarte[43].setBounds(x0y0.x + 450, x0y0.y, 55, 101);
```

```
    this.getContentPane().add(TavoloVerde);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    nuovaPartita();
}

// Metodo per caricare le immagini delle carte
private void caricaImmaginiCarte() {
    for (int i = 1; i < Carte.length; i++) {
        String imagePath = "Napoletane/" + Carte[i] + ".png";
        File file = new File(imagePath);
        if (!file.exists()) {
            System.out.println("Immagine non trovata: " + file.getAbsolutePath());
        }
        ImageIcon icon = new ImageIcon(imagePath);
        if (icon.getIconWidth() == -1) {
            System.out.println("Errore nel caricamento dell'immagine: " + imagePath);
        }
        imgCarte[i] = new JLabel(icon);
    }
}

public void Mescola() {
    for (int i = 39; i > 1; i--) {
        int x = (int)(Math.random() * i);
        int tmp = Mazzo[x];
        Mazzo[x] = Mazzo[i];
        Mazzo[i] = tmp;
    }
}

public void nuovaPartita() {
```

```
    for (int i = 1; i < 4; i++) {
        Carta[i].setEnabled(false);
        Sto[i].setEnabled(false);
        Punti[i] = 0;
        Punteggio[i].setText("0.0 (" + nomiGiocatori[i] + ")");
    }
    Punti[0] = 0;
    Punteggio[0].setText("0.0 (" + nomiGiocatori[0] + ")");
    Giocatore = 0;
    terminato = false;
    x0y0 = new Point(10, 25);

    Carta[0].setEnabled(true);
    Sto[0].setEnabled(true);
    nomeGiocatoreCorrente = nomiGiocatori[0];
}

public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();

    if (command.equals("Rinizia")) {
        restartGame();
        return;
    } else if (command.equals("Termina")) {
        quitGame();
        return;
    }

    if (terminato) return;

    if (command.startsWith("Carta")) {
        int playerIndex = Integer.parseInt(command.substring(5));
```

```
        handleCartaAction(playerIndex);
    } else if (command.startsWith("Sto")) {
        int playerIndex = Integer.parseInt(command.substring(3));
        handleStoAction(playerIndex);
    }
}

private void handleCartaAction(int playerIndex) {
    if (terminato) return;

    prosCarta++;
    String imagePath = "Napoletane/" + Carte[Mazzo[prosCarta]] + ".png";
    ImageIcon icon = new ImageIcon(imagePath);
    if (icon.getIconWidth() == -1) {
        System.out.println("Errore nel caricamento dell'immagine: " + imagePath);
    }
    imgCarte[prosCarta].setIcon(icon);
    TavoloVerde.add(imgCarte[prosCarta], new Integer(prosCarta));
    imgCarte[prosCarta].setBounds(x0y0.x, x0y0.y, 55, 101);
    x0y0.x += dxdy;
    x0y0.y += dxdy;

    int valore = Mazzo[prosCarta] % 10;
    if (valore > 7 || valore == 0) {
        Punti[Giocatore] += 0.5;
    } else {
        Punti[Giocatore] += valore;
    }
    Punteggio[Giocatore].setText(" " + Punti[Giocatore] + " (" + nomeGiocatoreCorrente + ")");

    if (Punti[Giocatore] > 7.5) {
```

```
            JOptionPane.showMessageDialog(this, "Hai sballato!", "Sballato",
JOptionPane.INFORMATION_MESSAGE);
            Avanza();
        } else if (Punti[Giocatore] == 7.5) {
            String vincitore = (Giocatore == 0) ? "Mazziere" : "Giocatore " + Giocatore;
            JOptionPane.showMessageDialog(this, vincitore + " ha vinto con 7.5!", "Vittoria",
JOptionPane.INFORMATION_MESSAGE);
            terminato = true;
            return;
        }

        // Se il giocatore è il giocatore 3, non terminare il suo turno finché non decide di terminarlo o
raggiunge 7.5 punti
        if (Giocatore == 3) {
            return;
        }

        // Controllo se tutti i giocatori hanno pescato
        boolean tuttiHannoPescato = true;
        for (double punti : Punti) {
            if (punti == 0) {
                tuttiHannoPescato = false;
                break;
            }
        }

        if (tuttiHannoPescato) {
            terminato = true; // Imposta il flag terminato per evitare che i giocatori pescino ulteriori carte

            double puntiVincenti = 0;
            int vincitoreIndex = -1;

            for (int i = 0; i < Punti.length; i++) {
```

```
            double differenza = Punti[i] - 7.5;
            // Se il giocatore ha sballato, salta
            if (differenza > 0) continue;
            // Se è più vicino a 7.5 del precedente vincitore, aggiorna il vincitore
            if (vincitoreIndex == -1 || Math.abs(differenza) < Math.abs(puntiVincenti)) {
                puntiVincenti = differenza;
                vincitoreIndex = i;
            }
        }

        if (vincitoreIndex != -1) {
            String vincitoreMsg;
            if (vincitoreIndex == 0) {
                vincitoreMsg = "Mazziere";
            } else {
                vincitoreMsg = "Giocatore " + vincitoreIndex;
            }
            JOptionPane.showMessageDialog(this, vincitoreMsg + " ha vinto per avvicinamento a 7.5!",
"Vittoria", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Nessun vincitore trovato!", "Errore",
JOptionPane.ERROR_MESSAGE);
        }
    }
}




private void handleStoAction(int playerIndex) {
    if (Giocatore != playerIndex) return;
    Avanza();
}
```

```
    public void Avanza() {
        Sto[Giocatore].setEnabled(false);
        Carta[Giocatore].setEnabled(false);

        Giocatore++;
        if (Giocatore == 4) {
            terminato = true;
            return;
        }

        Carta[Giocatore].setEnabled(true);
        Sto[Giocatore].setEnabled(true);
        x0y0 = new Point(10 + Giocatore * 150, 25);
        nomeGiocatoreCorrente = nomiGiocatori[Giocatore];
    }

    private void restartGame() {
        this.dispose();
        MezzoSette newGame = new MezzoSette();
        newGame.setTitle("Gioco del 7 e mezzo");
        newGame.setSize(600, 400);
        newGame.setVisible(true);
    }

    private void quitGame() {
        System.exit(0);
    }
}
```