```java
// QuizGame.java
import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.SwingConstants;
import javax.swing.Timer;

public class QuizGame extends JFrame {

    private ArrayList<Domanda> domande;
    private ArrayList<Boolean> risposteCorrette;
    private JPanel pnlCarte;
    private CardLayout cardLayout;
    private JButton btnAvanti, btnIndietro, btnRigioca;
    private JLabel lblPunteggio, lblTempo;
    private JProgressBar progTempo;
    private int domandaCorrente, punteggio, secondiRimanenti;
    private Timer timer;
    private TimerQuiz timerQuiz;

    public QuizGame() {
        super("Quiz a Tempo");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);

        // Inizializzazione delle variabili
        domande = new ArrayList<>();
        risposteCorrette = new ArrayList<>();
        domandaCorrente = 0;
        punteggio = 0;
        secondiRimanenti = 15; // 15 secondi per domanda
```

```
        // Inizializzazione componenti
        inizializzaDomande();
        initComponenti();
        initPannelli();
        initAscoltatori();

        this.setSize(600, 500);
        this.setVisible(true);

        // Avvia il timer
        avviaTimer();
    }

    private void inizializzaDomande() {
        // Aggiunge le domande al quiz
        ArrayList<String> risposte1 = new ArrayList<>();
        risposte1.add("8 bit");
        risposte1.add("16 bit");
        risposte1.add("32 bit");
        risposte1.add("64 bit");
        domande.add(new Domanda("Quanti bit ci sono in un byte?", risposte1,
0));

        ArrayList<String> risposte2 = new ArrayList<>();
        risposte2.add("Hyper Text Markup Language");
        risposte2.add("High Technology Modern Language");
        risposte2.add("Hyper Transfer Main Loop");
        risposte2.add("Home Tool Markup Language");
        domande.add(new Domanda("Cosa significa HTML?", risposte2, 0));

        ArrayList<String> risposte3 = new ArrayList<>();
        risposte3.add("Un insieme di istruzioni");
        risposte3.add("Una variabile di tipo intero");
        risposte3.add("Una classe astratta");
        risposte3.add("Un errore di compilazione");
        domande.add(new Domanda("Cos'è un algoritmo?", risposte3, 0));

        ArrayList<String> risposte4 = new ArrayList<>();
        risposte4.add("C");
        risposte4.add("C++");
        risposte4.add("Java");
        risposte4.add("Python");
        domande.add(new Domanda("Quale linguaggio è stato sviluppato da Sun
Microsystems?", risposte4, 2));

        ArrayList<String> risposte5 = new ArrayList<>();
        risposte5.add("Cascading Style Sheets");
        risposte5.add("Computer Style Sheets");
        risposte5.add("Creative Style Sheets");
        risposte5.add("Colorful Style Sheets");
```

```
        domande.add(new Domanda("Cosa significa CSS?", risposte5, 0));

        ArrayList<String> risposte6 = new ArrayList<>();
        risposte6.add("HyperText Transfer Protocol");
        risposte6.add("High Transfer Text Protocol");
        risposte6.add("Hyper Transfer Technology Process");
        risposte6.add("Home Transfer Text Protocol");
        domande.add(new Domanda("Cosa significa HTTP?", risposte6, 0));

        ArrayList<String> risposte7 = new ArrayList<>();
        risposte7.add("Linguaggio di programmazione");
        risposte7.add("Database relazionale");
        risposte7.add("Sistema operativo");
        risposte7.add("Formato di file");
        domande.add(new Domanda("Cos'è SQL?", risposte7, 1));

        ArrayList<String> risposte8 = new ArrayList<>();
        risposte8.add("Internet Protocol");
        risposte8.add("Internet Program");
        risposte8.add("Interface Protocol");
        risposte8.add("Internal Process");
        domande.add(new Domanda("Cosa significa IP?", risposte8, 0));

        ArrayList<String> risposte9 = new ArrayList<>();
        risposte9.add("Un'interfaccia grafica");
        risposte9.add("Un linguaggio di programmazione");
        risposte9.add("Un IDE per sviluppatori");
        risposte9.add("Una libreria grafica di Java");
        domande.add(new Domanda("Cos'è Swing in Java?", risposte9, 3));

        ArrayList<String> risposte10 = new ArrayList<>();
        risposte10.add("Spaghetti code");
        risposte10.add("Clean code");
        risposte10.add("Server code");
        risposte10.add("System code");
        domande.add(new Domanda("Come si chiama il codice difficile da
comprendere e manutenere?", risposte10, 0));

        // Inizializza array risposte
        for (int i = 0; i < domande.size(); i++) {
            risposteCorrette.add(false);
        }
    }

    private void initComponenti() {
        // Layout carte per le domande
        cardLayout = new CardLayout();
        pnlCarte = new JPanel(cardLayout);

        // Pulsanti navigazione
```

```
        btnAvanti = new JButton("Avanti >");
        btnAvanti.setFont(new Font("Sans-Serif", Font.BOLD, 14));

        btnIndietro = new JButton("< Indietro");
        btnIndietro.setFont(new Font("Sans-Serif", Font.BOLD, 14));
        btnIndietro.setEnabled(false);

        btnRigioca = new JButton("Nuova Partita");
        btnRigioca.setFont(new Font("Sans-Serif", Font.BOLD, 14));
        btnRigioca.setVisible(false);

        // Label e progress bar
        lblPunteggio = new JLabel("Punteggio: 0");
        lblPunteggio.setFont(new Font("Sans-Serif", Font.BOLD, 14));

        lblTempo = new JLabel("Tempo: " + secondiRimanenti + "s");
        lblTempo.setFont(new Font("Sans-Serif", Font.BOLD, 14));

        progTempo = new JProgressBar(0, secondiRimanenti);
        progTempo.setValue(secondiRimanenti);
        progTempo.setStringPainted(true);
        progTempo.setPreferredSize(new Dimension(200, 20));
        progTempo.setForeground(Color.GREEN);
    }

    private void initPannelli() {
        Container contentPane = this.getContentPane();
        contentPane.setLayout(new BorderLayout(10, 10));

        // Pannello superiore - Punteggio e timer
        JPanel pnlTop = new JPanel(new GridLayout(1, 2));
        pnlTop.setBorder(BorderFactory.createEmptyBorder(10, 10, 0, 10));

        JPanel pnlPunteggio = new JPanel(new FlowLayout(FlowLayout.LEFT));
        pnlPunteggio.add(lblPunteggio);

        JPanel pnlTimer = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        pnlTimer.add(lblTempo);
        pnlTimer.add(progTempo);

        pnlTop.add(pnlPunteggio);
        pnlTop.add(pnlTimer);

        // Pannello domande - Card Layout
        pnlCarte.setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createEmptyBorder(10, 10, 10, 10),
                BorderFactory.createLineBorder(Color.GRAY)));

        // Creazione delle carte per ogni domanda
        for (int i = 0; i < domande.size(); i++) {
```

```
            Domanda domanda = domande.get(i);
            DomandaPanel pnlDomanda = new DomandaPanel(domanda, i);
            pnlCarte.add(pnlDomanda, "Domanda" + i);

            // Aggiunge ascoltatore per le risposte
            pnlDomanda.setAscoltatore(new AscoltaRisposte(i));
        }

        // Pannello inferiore - Pulsanti navigazione
        JPanel pnlBottom = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
0));
        pnlBottom.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));

        pnlBottom.add(btnIndietro);
        pnlBottom.add(btnRigioca);
        pnlBottom.add(btnAvanti);

        // Aggiunta pannelli al contentPane
        contentPane.add(pnlTop, BorderLayout.NORTH);
        contentPane.add(pnlCarte, BorderLayout.CENTER);
        contentPane.add(pnlBottom, BorderLayout.SOUTH);
    }

    private void initAscoltatori() {
        // Ascoltatore 1: Classe anonima per i pulsanti di navigazione
        btnAvanti.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (domandaCorrente < domande.size() - 1) {
                    // Passa alla domanda successiva
                    domandaCorrente++;
                    cardLayout.show(pnlCarte, "Domanda" + domandaCorrente);
                    btnIndietro.setEnabled(true);

                    if (domandaCorrente == domande.size() - 1) {
                        btnAvanti.setText("Termina Quiz");
                    }
                } else {
                    // Fine del quiz
                    terminaQuiz();
                }

                // Resetta il timer
                resetTimer();
            }
        });

        btnIndietro.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
                if (domandaCorrente > 0) {
                    domandaCorrente--;
                    cardLayout.show(pnlCarte, "Domanda" + domandaCorrente);

                    if (domandaCorrente == 0) {
                        btnIndietro.setEnabled(false);
                    }

                    btnAvanti.setText("Avanti >");
                }

                // Resetta il timer
                resetTimer();
            }
        });

        btnRigioca.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Resetta il quiz
                domandaCorrente = 0;
                punteggio = 0;

                for (int i = 0; i < risposteCorrette.size(); i++) {
                    risposteCorrette.set(i, false);
                }

                lblPunteggio.setText("Punteggio: 0");

                // Resetta le domande
                for (int i = 0; i < domande.size(); i++) {
                    DomandaPanel pnl = (DomandaPanel)
pnlCarte.getComponent(i);
                    pnl.resetDomanda();
                }

                // Torna alla prima domanda
                cardLayout.show(pnlCarte, "Domanda0");

                // Aggiorna UI
                btnAvanti.setText("Avanti >");
                btnAvanti.setVisible(true);
                btnIndietro.setEnabled(false);
                btnRigioca.setVisible(false);

                // Resetta e avvia timer
                resetTimer();
                avviaTimer();
            }
        });
```

```
        // Ascoltatore 2: Classe esterna per il timer
        timerQuiz = new TimerQuiz(this);
        timer = new Timer(1000, timerQuiz);
    }

    // Metodi per gestire il timer
    public void avviaTimer() {
        timer.start();
    }

    public void fermaTimer() {
        timer.stop();
    }

    public void resetTimer() {
        secondiRimanenti = 15;
        lblTempo.setText("Tempo: " + secondiRimanenti + "s");
        progTempo.setValue(secondiRimanenti);
        progTempo.setForeground(Color.GREEN);

        // Se il timer era fermo, lo riavvia
        if (!timer.isRunning()) {
            timer.start();
        }
    }

    public void aggiornaTimer() {
        secondiRimanenti--;

        if (secondiRimanenti >= 0) {
            lblTempo.setText("Tempo: " + secondiRimanenti + "s");
            progTempo.setValue(secondiRimanenti);

            // Cambia colore in base al tempo rimanente
            if (secondiRimanenti <= 5) {
                progTempo.setForeground(Color.RED);
            } else if (secondiRimanenti <= 10) {
                progTempo.setForeground(Color.ORANGE);
            }
        } else {
            // Tempo scaduto
            fermaTimer();
            JOptionPane.showMessageDialog(this,
                    "Tempo scaduto per questa domanda!",
                    "Tempo Scaduto",
                    JOptionPane.WARNING_MESSAGE);

            // Passa alla domanda successiva
            btnAvanti.doClick();
```

```
        }
    }

    // Metodo per aggiornare il punteggio
    public void aggiornaPunteggio(int indiceDomanda, boolean corretta) {
        risposteCorrette.set(indiceDomanda, corretta);

        if (corretta) {
            // Calcola punteggio in base al tempo rimanente
            int puntiDomanda = 10 + secondiRimanenti;
            punteggio += puntiDomanda;
            lblPunteggio.setText("Punteggio: " + punteggio);
        }
    }

    // Metodo per terminare il quiz
    private void terminaQuiz() {
        fermaTimer();

        // Conta risposte corrette
        int corrette = 0;
        for (boolean risposta : risposteCorrette) {
            if (risposta) corrette++;
        }

        // Mostra riepilogo
        JOptionPane.showMessageDialog(this,
                "Quiz completato!\n\n" +
                "Risposte corrette: " + corrette + "/" + domande.size() +
"\n" +
                "Punteggio totale: " + punteggio,
                "Quiz Terminato",
                JOptionPane.INFORMATION_MESSAGE);

        // Aggiorna UI
        btnAvanti.setVisible(false);
        btnRigioca.setVisible(true);
    }

    // Classe interna per l'ascoltatore delle risposte
    private class AscoltaRisposte implements ActionListener {
        private int indiceDomanda;

        public AscoltaRisposte(int indiceDomanda) {
            this.indiceDomanda = indiceDomanda;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            int risposta = Integer.parseInt(e.getActionCommand());
```

```
            Domanda domanda = domande.get(indiceDomanda);

            boolean corretta = (risposta == domanda.getRispostaCorretta());
            DomandaPanel panel = (DomandaPanel)
pnlCarte.getComponent(indiceDomanda);

            // Mostra feedback
            panel.mostraRisultato(risposta, corretta);

            // Aggiorna punteggio
            aggiornaPunteggio(indiceDomanda, corretta);

            // Ferma il timer per questa domanda
            fermaTimer();
        }
    }

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

// Domanda.java
import java.util.ArrayList;

public class Domanda {
    private String testoDomanda;
    private ArrayList<String> risposte;
    private int rispostaCorretta;

    public Domanda(String testoDomanda, ArrayList<String> risposte, int
rispostaCorretta) {
        this.testoDomanda = testoDomanda;
        this.risposte = risposte;
        this.rispostaCorretta = rispostaCorretta;
    }

    public String getTestoDomanda() {
        return testoDomanda;
    }

    public ArrayList<String> getRisposte() {
        return risposte;
    }

    public int getRispostaCorretta() {
        return rispostaCorretta;
    }
}
```

```java
// DomandaPanel.java
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.SwingConstants;

public class DomandaPanel extends JPanel {

    private Domanda domanda;
    private JLabel lblDomanda, lblNumero;
    private JRadioButton[] radioBtnRisposte;
    private ButtonGroup btnGroup;
    private int indice;

    public DomandaPanel(Domanda domanda, int indice) {
        super(new BorderLayout(10, 10));
        this.domanda = domanda;
        this.indice = indice;

        // Impostazioni pannello
        this.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        // Inizializzazione componenti
        initComponenti();
    }

    private void initComponenti() {
        // Label numero domanda
        lblNumero = new JLabel("Domanda " + (indice + 1));
        lblNumero.setFont(new Font("Sans-Serif", Font.BOLD, 14));

        // Label testo domanda
        lblDomanda = new JLabel(domanda.getTestoDomanda());
        lblDomanda.setFont(new Font("Sans-Serif", Font.BOLD, 18));
        lblDomanda.setHorizontalAlignment(SwingConstants.CENTER);
        lblDomanda.setBorder(BorderFactory.createEmptyBorder(10, 0, 20, 0));

        // Radio buttons per le risposte
        ArrayList<String> risposte = domanda.getRisposte();
        radioBtnRisposte = new JRadioButton[risposte.size()];
        btnGroup = new ButtonGroup();
```

```
        // Pannello risposte
        JPanel pnlRisposte = new JPanel(new GridLayout(risposte.size(), 1,
0, 5));
        pnlRisposte.setBorder(BorderFactory.createEmptyBorder(0, 20, 0,
20));

        // Creazione radio buttons
        for (int i = 0; i < risposte.size(); i++) {
            radioBtnRisposte[i] = new JRadioButton(risposte.get(i));
            radioBtnRisposte[i].setFont(new Font("Sans-Serif", Font.PLAIN,
16));
            radioBtnRisposte[i].setActionCommand(String.valueOf(i));

            btnGroup.add(radioBtnRisposte[i]);
            pnlRisposte.add(radioBtnRisposte[i]);
        }

        // Aggiunta componenti al pannello
        this.add(lblNumero, BorderLayout.NORTH);
        this.add(lblDomanda, BorderLayout.CENTER);
        this.add(pnlRisposte, BorderLayout.SOUTH);
    }

    // Imposta l'ascoltatore per le risposte
    public void setAscoltatore(ActionListener ascoltatore) {
        for (JRadioButton radioBtn : radioBtnRisposte) {
            radioBtn.addActionListener(ascoltatore);
        }
    }

    // Mostra il risultato della risposta
    public void mostraRisultato(int rispostaData, boolean corretta) {
        // Disabilita tutte le risposte
        for (JRadioButton radioBtn : radioBtnRisposte) {
            radioBtn.setEnabled(false);
        }

        // Evidenzia la risposta data
        radioBtnRisposte[rispostaData].setForeground(corretta ? Color.GREEN
: Color.RED);
        radioBtnRisposte[rispostaData].setFont(new Font("Sans-Serif",
Font.BOLD, 16));

        // Se la risposta è sbagliata, evidenzia anche quella corretta
        if (!corretta) {

radioBtnRisposte[domanda.getRispostaCorretta()].setForeground(Color.GREEN);
            radioBtnRisposte[domanda.getRispostaCorretta()].setFont(new
Font("Sans-Serif", Font.BOLD, 16));
        }
```

```
    }

    // Resetta il pannello per un nuovo quiz
    public void resetDomanda() {
        // Riabilita tutte le risposte
        for (JRadioButton radioBtn : radioBtnRisposte) {
            radioBtn.setEnabled(true);
            radioBtn.setForeground(Color.BLACK);
            radioBtn.setFont(new Font("Sans-Serif", Font.PLAIN, 16));
            radioBtn.setSelected(false);
        }
    }
}

// TimerQuiz.java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TimerQuiz implements ActionListener {

    private QuizGame quizGame;

    public TimerQuiz(QuizGame quizGame) {
        this.quizGame = quizGame;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // Aggiorna il timer ad ogni tick
        quizGame.aggiornaTimer();
    }
}
```