Componenti e Metodi Fondamentali

Struttura Base di un'Applicazione Swing

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

public class AppBase extends JFrame {
    public AppBase() {
        // Configurazione della finestra
        setTitle("Titolo Applicazione");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null); // Centra la finestra

        // Aggiunta componenti
        // ...

        // Rendi visibile la finestra
        setVisible(true);
    }

    public static void main(String[] args) {
        // Avvio dell'applicazione sull'Event Dispatch Thread
        SwingUtilities.invokeLater(() -> new AppBase());
    }
}
```

Container Principali

- `JFrame` : Finestra principale dell'applicazione
- `JPanel` : Contenitore per organizzare componenti
- `JDialog` : Finestra di dialogo
- `JTabbedPane` : Contenitore a schede

Componenti Base

- `JLabel` : Etichetta di testo/immagine
- `JButton` : Pulsante cliccabile
- `JTextField` : Campo di testo a riga singola
- `JTextArea` : Area di testo multi-riga
- `JCheckBox` : Casella di controllo

- `JRadioButton` : Pulsante di opzione
- `JComboBox` : Menu a tendina
- `JList` : Lista selezionabile
- `JTable` : Tabella dati
- `JScrollPane` : Pannello con barre di scorrimento

Layout Manager

- `FlowLayout` : Dispone componenti in fila
- `BorderLayout` : Dispone in 5 aree (NORTH, SOUTH, EAST, WEST, CENTER)
- `GridLayout` : Dispone in griglia di righe e colonne
- `BoxLayout` : Dispone in riga o colonna
- `GridBagLayout` : Layout flessibile e potente

Eventi Principali

- `ActionListener` : Per eventi di clic su pulsanti, selezione da menu, ecc.
- `MouseListener` : Per eventi del mouse (clic, movimento, ecc.)
- `KeyListener` : Per eventi da tastiera
- `WindowListener` : Per eventi della finestra (chiusura, ecc.)
- `ItemListener` : Per eventi di selezione (checkbox, combobox, ecc.)

Gestione Eventi (esempio con ActionListener)

```
JButton button = new JButton("Cliccami");
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Codice da eseguire quando il pulsante viene cliccato
        JOptionPane.showMessageDialog(null, "Pulsante cliccato!");
    }
});

// Oppure con lambda (Java 8+)
button.addActionListener(e -> JOptionPane.showMessageDialog(null, "Pulsante
cliccato!"));
```

Esercizi Risolti

Esercizio 1: Calcolatrice Semplice

Una calcolatrice base con operazioni di addizione, sottrazione, moltiplicazione e divisione.

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

public class Calcolatrice extends JFrame {
    private JTextField display;
    private double num1 = 0, num2 = 0;
    private String operazione = "";

    public Calcolatrice() {
        setTitle("Calcolatrice");
        setSize(300, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Pannello display
        JPanel panDisplay = new JPanel();
        display = new JTextField(15);
        display.setEditable(false);
        display.setHorizontalAlignment(JTextField.RIGHT);
        panDisplay.add(display);

        // Pannello pulsanti
        JPanel panPulsanti = new JPanel();
        panPulsanti.setLayout(new GridLayout(4, 4, 5, 5));

        // Pulsanti numerici
        for (int i = 1; i <= 9; i++) {
            JButton btn = new JButton(String.valueOf(i));
            btn.addActionListener(e -> display.setText(display.getText() +
((JButton)e.getSource()).getText()));
            panPulsanti.add(btn);
        }

        // Pulsanti operazioni
        JButton btnPiu = new JButton("+");
        btnPiu.addActionListener(e -> {
            num1 = Double.parseDouble(display.getText());
            operazione = "+";
            display.setText("");
        });

        JButton btnMeno = new JButton("-");
        btnMeno.addActionListener(e -> {
            num1 = Double.parseDouble(display.getText());
            operazione = "-";
            display.setText("");
        });
```

```
        JButton btnMoltiplica = new JButton("*");
        btnMoltiplica.addActionListener(e -> {
            num1 = Double.parseDouble(display.getText());
            operazione = "*";
            display.setText("");
        });

        JButton btnDividi = new JButton("/");
        btnDividi.addActionListener(e -> {
            num1 = Double.parseDouble(display.getText());
            operazione = "/";
            display.setText("");
        });

        JButton btnZero = new JButton("0");
        btnZero.addActionListener(e -> display.setText(display.getText() +
"0"));

        JButton btnUguale = new JButton("=");
        btnUguale.addActionListener(e -> {
            num2 = Double.parseDouble(display.getText());
            double risultato = 0;

            switch (operazione) {
                case "+":
                    risultato = num1 + num2;
                    break;
                case "-":
                    risultato = num1 - num2;
                    break;
                case "*":
                    risultato = num1 * num2;
                    break;
                case "/":
                    if (num2 != 0) {
                        risultato = num1 / num2;
                    } else {
                        JOptionPane.showMessageDialog(null, "Divisione per
zero non consentita");
                        display.setText("");
                        return;
                    }
                    break;
            }

            display.setText(String.valueOf(risultato));
        });

        JButton btnCancella = new JButton("C");
        btnCancella.addActionListener(e -> {
```

```
            display.setText("");
            num1 = 0;
            num2 = 0;
            operazione = "";
        });

        // Aggiungi pulsanti al pannello
        panPulsanti.add(btnPiu);
        panPulsanti.add(btnMeno);
        panPulsanti.add(btnMoltiplica);
        panPulsanti.add(btnDividi);
        panPulsanti.add(btnZero);
        panPulsanti.add(btnUguale);
        panPulsanti.add(btnCancella);

        // Layout generale
        setLayout(new BorderLayout());
        add(panDisplay, BorderLayout.NORTH);
        add(panPulsanti, BorderLayout.CENTER);

        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new Calcolatrice());
    }
}
```

Esercizio 2: Form di Registrazione

Un form per inserire dati personali con validazione.

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

public class FormRegistrazione extends JFrame {
    private JTextField txtNome, txtCognome, txtEmail;
    private JPasswordField txtPassword;
    private JComboBox<String> cmbCitta;
    private JRadioButton rbMaschio, rbFemmina;
    private JCheckBox chkAccetto;

    public FormRegistrazione() {
        setTitle("Form di Registrazione");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
```

```
        // Layout principale
        setLayout(new GridLayout(8, 1, 10, 10));

        // Nome
        JPanel panNome = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panNome.add(new JLabel("Nome:"));
        txtNome = new JTextField(20);
        panNome.add(txtNome);
        add(panNome);

        // Cognome
        JPanel panCognome = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panCognome.add(new JLabel("Cognome:"));
        txtCognome = new JTextField(20);
        panCognome.add(txtCognome);
        add(panCognome);

        // Email
        JPanel panEmail = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panEmail.add(new JLabel("Email:"));
        txtEmail = new JTextField(20);
        panEmail.add(txtEmail);
        add(panEmail);

        // Password
        JPanel panPassword = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panPassword.add(new JLabel("Password:"));
        txtPassword = new JPasswordField(20);
        panPassword.add(txtPassword);
        add(panPassword);

        // Città
        JPanel panCitta = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panCitta.add(new JLabel("Città:"));
        String[] citta = {"Roma", "Milano", "Napoli", "Torino", "Palermo"};
        cmbCitta = new JComboBox<>(citta);
        panCitta.add(cmbCitta);
        add(panCitta);

        // Genere
        JPanel panGenere = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panGenere.add(new JLabel("Genere:"));
        rbMaschio = new JRadioButton("Maschio");
        rbFemmina = new JRadioButton("Femmina");
        ButtonGroup bgGenere = new ButtonGroup();
        bgGenere.add(rbMaschio);
        bgGenere.add(rbFemmina);
        panGenere.add(rbMaschio);
        panGenere.add(rbFemmina);
        add(panGenere);
```

```
        // Termini e condizioni
        JPanel panTermini = new JPanel(new FlowLayout(FlowLayout.LEFT));
        chkAccetto = new JCheckBox("Accetto i termini e le condizioni");
        panTermini.add(chkAccetto);
        add(panTermini);

        // Pulsanti
        JPanel panPulsanti = new JPanel(new FlowLayout(FlowLayout.CENTER));
        JButton btnRegistra = new JButton("Registra");
        JButton btnAnnulla = new JButton("Annulla");

        btnRegistra.addActionListener(e -> {
            if (validateForm()) {
                JOptionPane.showMessageDialog(null, "Registrazione
completata con successo!");
                clearForm();
            }
        });

        btnAnnulla.addActionListener(e -> clearForm());

        panPulsanti.add(btnRegistra);
        panPulsanti.add(btnAnnulla);
        add(panPulsanti);

        setVisible(true);
    }

    private boolean validateForm() {
        if (txtNome.getText().trim().isEmpty()) {
            JOptionPane.showMessageDialog(this, "Inserire il nome",
"Errore", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        if (txtCognome.getText().trim().isEmpty()) {
            JOptionPane.showMessageDialog(this, "Inserire il cognome",
"Errore", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        String email = txtEmail.getText().trim();
        if (email.isEmpty() || !email.contains("@") || !email.contains("."))
{
            JOptionPane.showMessageDialog(this, "Inserire un'email valida",
"Errore", JOptionPane.ERROR_MESSAGE);
            return false;
        }
```

```
        if (new String(txtPassword.getPassword()).length() < 6) {
            JOptionPane.showMessageDialog(this, "La password deve essere di
almeno 6 caratteri", "Errore", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        if (!rbMaschio.isSelected() && !rbFemmina.isSelected()) {
            JOptionPane.showMessageDialog(this, "Selezionare il genere",
"Errore", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        if (!chkAccetto.isSelected()) {
            JOptionPane.showMessageDialog(this, "Accettare i termini e le
condizioni", "Errore", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        return true;
    }

    private void clearForm() {
        txtNome.setText("");
        txtCognome.setText("");
        txtEmail.setText("");
        txtPassword.setText("");
        cmbCitta.setSelectedIndex(0);
        rbMaschio.setSelected(false);
        rbFemmina.setSelected(false);
        chkAccetto.setSelected(false);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new FormRegistrazione());
    }
}
```

Esercizio 3: Lista della Spesa

Un'applicazione per gestire una lista della spesa.

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

public class ListaSpesa extends JFrame {
    private DefaultListModel<String> modelLista;
    private JList<String> lista;
```

```
    private JTextField txtProdotto;
    private ArrayList<String> prodotti = new ArrayList<>();

    public ListaSpesa() {
        setTitle("Lista della Spesa");
        setSize(400, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Panel per inserimento prodotti
        JPanel panInserimento = new JPanel();
        JLabel lblProdotto = new JLabel("Prodotto:");
        txtProdotto = new JTextField(15);
        JButton btnAggiungi = new JButton("Aggiungi");

        panInserimento.add(lblProdotto);
        panInserimento.add(txtProdotto);
        panInserimento.add(btnAggiungi);

        // Panel per la lista
        JPanel panLista = new JPanel(new BorderLayout());
        modelLista = new DefaultListModel<>();
        lista = new JList<>(modelLista);
        JScrollPane scrollPane = new JScrollPane(lista);
        panLista.add(new JLabel("Lista della spesa:"), BorderLayout.NORTH);
        panLista.add(scrollPane, BorderLayout.CENTER);

        // Panel per i pulsanti di gestione
        JPanel panGestione = new JPanel();
        JButton btnRimuovi = new JButton("Rimuovi");
        JButton btnPulisci = new JButton("Pulisci lista");
        JButton btnSalva = new JButton("Salva lista");

        panGestione.add(btnRimuovi);
        panGestione.add(btnPulisci);
        panGestione.add(btnSalva);

        // Aggiungi event listeners
        btnAggiungi.addActionListener(e -> aggiungiProdotto());

        btnRimuovi.addActionListener(e -> {
            int selectedIndex = lista.getSelectedIndex();
            if (selectedIndex != -1) {
                prodotti.remove(selectedIndex);
                modelLista.remove(selectedIndex);
            } else {
                JOptionPane.showMessageDialog(null, "Seleziona un prodotto
da rimuovere");
            }
        });
```

```
        btnPulisci.addActionListener(e -> {
            prodotti.clear();
            modelLista.clear();
        });

        btnSalva.addActionListener(e -> {
            if (prodotti.isEmpty()) {
                JOptionPane.showMessageDialog(null, "La lista è vuota!");
            } else {
                StringBuilder sb = new StringBuilder("Lista della spesa
salvata:\n");
                for (String prodotto : prodotti) {
                    sb.append("- ").append(prodotto).append("\n");
                }
                JOptionPane.showMessageDialog(null, sb.toString());
            }
        });

        // Aggiungi key listener al campo di testo
        txtProdotto.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                    aggiungiProdotto();
                }
            }
        });

        // Layout generale
        setLayout(new BorderLayout(10, 10));
        add(panInserimento, BorderLayout.NORTH);
        add(panLista, BorderLayout.CENTER);
        add(panGestione, BorderLayout.SOUTH);

        // Padding

((JPanel)getContentPane()).setBorder(BorderFactory.createEmptyBorder(10, 10,
10, 10));

        setVisible(true);
    }

    private void aggiungiProdotto() {
        String prodotto = txtProdotto.getText().trim();
        if (!prodotto.isEmpty()) {
            prodotti.add(prodotto);
            modelLista.addElement(prodotto);
            txtProdotto.setText("");
            txtProdotto.requestFocus();
```

```
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ListaSpesa());
    }
}
```

Esercizio 4: Editor di Testo Semplice

Un semplice editor di testo con funzionalità di base.

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

public class EditorTesto extends JFrame {
    private JTextArea areaEditor;
    private JFileChooser fileChooser;
    private File fileCorrente;

    public EditorTesto() {
        setTitle("Editor di Testo");
        setSize(600, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Menu bar
        JMenuBar menuBar = new JMenuBar();

        // Menu File
        JMenu menuFile = new JMenu("File");
        JMenuItem miNuovo = new JMenuItem("Nuovo");
        JMenuItem miApri = new JMenuItem("Apri");
        JMenuItem miSalva = new JMenuItem("Salva");
        JMenuItem miSalvaConNome = new JMenuItem("Salva con nome");
        JMenuItem miEsci = new JMenuItem("Esci");

        menuFile.add(miNuovo);
        menuFile.add(miApri);
        menuFile.add(miSalva);
        menuFile.add(miSalvaConNome);
        menuFile.addSeparator();
        menuFile.add(miEsci);

        // Menu Modifica
        JMenu menuModifica = new JMenu("Modifica");
        JMenuItem miTaglia = new JMenuItem("Taglia");
```

```
        JMenuItem miCopia = new JMenuItem("Copia");
        JMenuItem miIncolla = new JMenuItem("Incolla");
        JMenuItem miSelezionaTutto = new JMenuItem("Seleziona tutto");

        menuModifica.add(miTaglia);
        menuModifica.add(miCopia);
        menuModifica.add(miIncolla);
        menuModifica.addSeparator();
        menuModifica.add(miSelezionaTutto);

        // Menu Formato
        JMenu menuFormato = new JMenu("Formato");
        JMenuItem miWordWrap = new JMenuItem("A capo automatico");
        JCheckBoxMenuItem cbWordWrap = new JCheckBoxMenuItem("A capo
automatico");
        menuFormato.add(cbWordWrap);

        // Aggiungi menu alla barra
        menuBar.add(menuFile);
        menuBar.add(menuModifica);
        menuBar.add(menuFormato);

        // Area di testo
        areaEditor = new JTextArea();
        areaEditor.setFont(new Font("Monospaced", Font.PLAIN, 14));
        JScrollPane scrollPane = new JScrollPane(areaEditor);

        // File chooser
        fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new
javax.swing.filechooser.FileNameExtensionFilter("File di testo (*.txt)",
"txt"));

        // Event listeners per i menu
        // File > Nuovo
        miNuovo.addActionListener(e -> {
            if (areaEditor.getText().length() > 0) {
                int risposta = JOptionPane.showConfirmDialog(this,
                    "Vuoi salvare il documento corrente?",
                    "Salva",
                    JOptionPane.YES_NO_CANCEL_OPTION);

                if (risposta == JOptionPane.YES_OPTION) {
                    if (salvaFile()) {
                        nuovoFile();
                    }
                } else if (risposta == JOptionPane.NO_OPTION) {
                    nuovoFile();
                }
            } else {
```

```
                nuovoFile();
            }
        });

        // File > Apri
        miApri.addActionListener(e -> {
            if (fileChooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
                apriFile(fileChooser.getSelectedFile());
            }
        });

        // File > Salva
        miSalva.addActionListener(e -> salvaFile());

        // File > Salva con nome
        miSalvaConNome.addActionListener(e -> salvaFileConNome());

        // File > Esci
        miEsci.addActionListener(e -> {
            if (areaEditor.getText().length() > 0) {
                int risposta = JOptionPane.showConfirmDialog(this,
                    "Vuoi salvare prima di uscire?",
                    "Salva",
                    JOptionPane.YES_NO_CANCEL_OPTION);

                if (risposta == JOptionPane.YES_OPTION) {
                    if (salvaFile()) {
                        System.exit(0);
                    }
                } else if (risposta == JOptionPane.NO_OPTION) {
                    System.exit(0);
                }
            } else {
                System.exit(0);
            }
        });

        // Modifica > Taglia
        miTaglia.addActionListener(e -> areaEditor.cut());

        // Modifica > Copia
        miCopia.addActionListener(e -> areaEditor.copy());

        // Modifica > Incolla
        miIncolla.addActionListener(e -> areaEditor.paste());

        // Modifica > Seleziona tutto
        miSelezionaTutto.addActionListener(e -> areaEditor.selectAll());
```

```
        // Formato > A capo automatico
        cbWordWrap.addActionListener(e -> {
            boolean isSelected = cbWordWrap.isSelected();
            areaEditor.setLineWrap(isSelected);
            areaEditor.setWrapStyleWord(isSelected);
        });

        // Layout generale
        setJMenuBar(menuBar);
        add(scrollPane);

        setVisible(true);
    }

    private void nuovoFile() {
        areaEditor.setText("");
        fileCorrente = null;
        setTitle("Editor di Testo - Nuovo documento");
    }

    private void apriFile(File file) {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader(file));
            areaEditor.setText("");
            String line;
            while ((line = reader.readLine()) != null) {
                areaEditor.append(line + "\n");
            }
            reader.close();
            fileCorrente = file;
            setTitle("Editor di Testo - " + file.getName());
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
                "Errore nell'apertura del file: " + e.getMessage(),
                "Errore",
                JOptionPane.ERROR_MESSAGE);
        }
    }

    private boolean salvaFile() {
        if (fileCorrente == null) {
            return salvaFileConNome();
        } else {
            return salvaFile(fileCorrente);
        }
    }

    private boolean salvaFileConNome() {
        if (fileChooser.showSaveDialog(this) == JFileChooser.APPROVE_OPTION)
```

```
{
            File file = fileChooser.getSelectedFile();
            // Aggiungi estensione .txt se necessario
            if (!file.getName().toLowerCase().endsWith(".txt")) {
                file = new File(file.getAbsolutePath() + ".txt");
            }
            return salvaFile(file);
        }
        return false;
    }

    private boolean salvaFile(File file) {
        try {
            BufferedWriter writer = new BufferedWriter(new
FileWriter(file));
            writer.write(areaEditor.getText());
            writer.close();
            fileCorrente = file;
            setTitle("Editor di Testo - " + file.getName());
            return true;
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
                "Errore nel salvataggio del file: " + e.getMessage(),
                "Errore",
                JOptionPane.ERROR_MESSAGE);
            return false;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new EditorTesto());
    }
}
```

Esercizio 5: Convertitore di Valute

Un'applicazione per convertire tra diverse valute.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.DecimalFormat;

public class ConvertitoreValute extends JFrame {
    private JTextField txtImporto;
    private JComboBox<String> cmbValutaOrigine, cmbValutaDestinazione;
    private JLabel lblRisultato;

    // Tassi di cambio rispetto all'euro (esempio)
```

```
    private final double[] TASSI = {1.0, 1.09, 0.86, 1.63, 138.0}; // EUR, USD, GBP, CAD, JPY
    private final String[] VALUTE = {"Euro (EUR)", "Dollaro (USD)", "Sterlina (GBP)", "Dollaro Canadese (CAD)", "Yen (JPY)"};

    public ConvertitoreValute() {
        setTitle("Convertitore di Valute");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Panel superiore per inserimento dati
        JPanel panInserimento = new JPanel(new GridLayout(3, 2, 10, 10));

        // Importo
        panInserimento.add(new JLabel("Importo:"));
        txtImporto = new JTextField();
        panInserimento.add(txtImporto);

        // Valuta di origine
        panInserimento.add(new JLabel("Da:"));
        cmbValutaOrigine = new JComboBox<>(VALUTE);
        panInserimento.add(cmbValutaOrigine);

        // Valuta di destinazione
        panInserimento.add(new JLabel("A:"));
        cmbValutaDestinazione = new JComboBox<>(VALUTE);
        cmbValutaDestinazione.setSelectedIndex(1); // Default USD
        panInserimento.add(cmbValutaDestinazione);

        // Panel centrale per risultato
        JPanel panRisultato = new JPanel();
        lblRisultato = new JLabel("Inserisci un importo e premi 'Converti'");
        lblRisultato.setFont(new Font("Arial", Font.BOLD, 14));
        panRisultato.add(lblRisultato);

        // Panel inferiore per pulsanti
        JPanel panPulsanti = new JPanel();
        JButton btnConverti = new JButton("Converti");
        JButton btnScambia = new JButton("Scambia valute");
        JButton btnPulisci = new JButton("Pulisci");

        panPulsanti.add(btnConverti);
        panPulsanti.add(btnScambia);
        panPulsanti.add(btnPulisci);

        // Event listeners
        btnConverti.addActionListener(e -> convertiValuta());
```

```
        btnScambia.addActionListener(e -> {
            int temp = cmbValutaOrigine.getSelectedIndex();

cmbValutaOrigine.setSelectedIndex(cmbValutaDestinazione.getSelectedIndex());
            cmbValutaDestinazione.setSelectedIndex(temp);
        });

        btnPulisci.addActionListener(e -> {
            txtImporto.setText("");
            lblRisultato.setText("Inserisci un importo e premi 'Converti'");
        });

        // Key listener per permettere la conversione con Enter
        txtImporto.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                    convertiValuta();
                }
            }
        });

        // Layout generale
        setLayout(new BorderLayout(20, 20));
        add(panInserimento, BorderLayout.NORTH);
        add(panRisultato, BorderLayout.CENTER);
        add(panPulsanti, BorderLayout.SOUTH);

        // Padding

((JPanel)getContentPane()).setBorder(BorderFactory.createEmptyBorder(20, 20,
20, 20));

        setVisible(true);
    }

private void convertiValuta() { try { double importo =
Double.parseDouble(txtImporto.getText().replace(",", ".")); int
valutaOrigine = cmbValutaOrigine.getSelectedIndex(); int valutaDestinazione
= cmbValutaDestinazione.getSelectedIndex();
```

```
    // Converti in euro come valuta intermedia
    double importoInEuro = importo / TASSI[valutaOrigine];

    // Converti da euro alla valuta di destinazione
    double risultato = importoInEuro * TASSI[valutaDestinazione];
```

```
    // Formatta il risultato
    DecimalFormat df = new DecimalFormat("#,##0.00");
    String valutaDest = VALUTE[valutaDestinazione].substring(0,
VALUTE[valutaDestinazione].indexOf(" "));
    lblRisultato.setText(df.format(importo) + " " +
VALUTE[valutaOrigine].substring(0, VALUTE[valutaOrigine].indexOf(" ")) +
                         " = " + df.format(risultato) + " " + valutaDest);
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(this, "Inserire un importo valido",
"Errore", JOptionPane.ERROR_MESSAGE);
}
```

}

public static void main(String[] args) {
SwingUtilities.invokeLater(() -> new ConvertitoreValute());
}

```
}
```

Esercizio 6: Gioco del Snake Semplificato

Una versione semplificata del classico gioco Snake.

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

public class GiocoSnake extends JFrame {
    private final int LARGHEZZA = 400;
    private final int ALTEZZA = 400;
    private final int DIMENSIONE_UNITA = 20;
    private final int UNITA_GIOCO = (LARGHEZZA * ALTEZZA) /
(DIMENSIONE_UNITA * DIMENSIONE_UNITA);
    private final int RITARDO = 150;

    private final int[] x = new int[UNITA_GIOCO];
    private final int[] y = new int[UNITA_GIOCO];

    private int lunghezzaSerpe = 3;
    private int melaX;
    private int melaY;
```

```
    private int punteggio = 0;

    private char direzione = 'D'; // Inizia verso destra (R = right)
    private boolean inGioco = false;

    private Timer timer;
    private Random random;
    private JButton btnNuovaPartita;

    public GiocoSnake() {
        setTitle("Snake");
        setSize(LARGHEZZA, ALTEZZA);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);

        random = new Random();

        // Pannello di gioco
        JPanel panGioco = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                disegnaGioco(g);
            }
        };
        panGioco.setBackground(Color.BLACK);

        // Pulsante nuova partita
        btnNuovaPartita = new JButton("Nuova Partita");
        btnNuovaPartita.addActionListener(e -> iniziaGioco());

        // Aggiungi key listener
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (!inGioco) {
                    if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                        iniziaGioco();
                    }
                    return;
                }

                switch (e.getKeyCode()) {
                    case KeyEvent.VK_LEFT:
                        if (direzione != 'D') {
                            direzione = 'A'; // A = sinistra
                        }
                        break;
                    case KeyEvent.VK_RIGHT:
```

```
                    if (direzione != 'A') {
                        direzione = 'D'; // D = destra
                    }
                    break;
                case KeyEvent.VK_UP:
                    if (direzione != 'S') {
                        direzione = 'W'; // W = su
                    }
                    break;
                case KeyEvent.VK_DOWN:
                    if (direzione != 'W') {
                        direzione = 'S'; // S = giù
                    }
                    break;
            }
        }
    });

    // Layout
    setLayout(new BorderLayout());
    add(panGioco, BorderLayout.CENTER);
    add(btnNuovaPartita, BorderLayout.SOUTH);

    setFocusable(true);
    setVisible(true);
}

private void iniziaGioco() {
    lunghezzaSerpe = 3;
    punteggio = 0;
    direzione = 'D';

    // Inizializza posizione serpe
    for (int i = 0; i < lunghezzaSerpe; i++) {
        x[i] = DIMENSIONE_UNITA * (5 - i);
        y[i] = DIMENSIONE_UNITA * 5;
    }

    // Genera prima mela
    generaMela();

    inGioco = true;
    btnNuovaPartita.setEnabled(false);

    // Avvia timer
    if (timer != null) {
        timer.stop();
    }

    timer = new Timer(RITARDO, e -> {
```

```
            if (inGioco) {
                muovi();
                controllaCollisioni();
                repaint();
            } else {
                timer.stop();
                btnNuovaPartita.setEnabled(true);
            }
        });

        timer.start();
    }

    private void generaMela() {
        melaX = random.nextInt((LARGHEZZA / DIMENSIONE_UNITA)) *
DIMENSIONE_UNITA;
        melaY = random.nextInt((ALTEZZA / DIMENSIONE_UNITA) - 1) *
DIMENSIONE_UNITA;
    }

    private void muovi() {
        for (int i = lunghezzaSerpe; i > 0; i--) {
            x[i] = x[i - 1];
            y[i] = y[i - 1];
        }

        switch (direzione) {
            case 'A': // Sinistra
                x[0] -= DIMENSIONE_UNITA;
                break;
            case 'D': // Destra
                x[0] += DIMENSIONE_UNITA;
                break;
            case 'W': // Su
                y[0] -= DIMENSIONE_UNITA;
                break;
            case 'S': // Giù
                y[0] += DIMENSIONE_UNITA;
                break;
        }
    }

    private void controllaCollisioni() {
        // Controlla collisione con se stesso
        for (int i = lunghezzaSerpe; i > 0; i--) {
            if (x[0] == x[i] && y[0] == y[i]) {
                inGioco = false;
                return;
            }
        }
```

```
        // Controlla collisione con i bordi
        if (x[0] < 0 || x[0] >= LARGHEZZA || y[0] < 0 || y[0] >= ALTEZZA -
DIMENSIONE_UNITA) {
            inGioco = false;
            return;
        }

        // Controlla collisione con la mela
        if (x[0] == melaX && y[0] == melaY) {
            lunghezzaSerpe++;
            punteggio++;
            generaMela();
        }
    }

    private void disegnaGioco(Graphics g) {
        if (inGioco) {
            // Disegna mela
            g.setColor(Color.RED);
            g.fillOval(melaX, melaY, DIMENSIONE_UNITA, DIMENSIONE_UNITA);

            // Disegna serpe
            for (int i = 0; i < lunghezzaSerpe; i++) {
                if (i == 0) {
                    g.setColor(Color.GREEN);
                } else {
                    g.setColor(new Color(0, 200, 0));
                }
                g.fillRect(x[i], y[i], DIMENSIONE_UNITA, DIMENSIONE_UNITA);
            }

            // Disegna punteggio
            g.setColor(Color.WHITE);
            g.setFont(new Font("Arial", Font.BOLD, 16));
            g.drawString("Punteggio: " + punteggio, 10, 20);
        } else {
            // Schermata game over
            g.setColor(Color.RED);
            g.setFont(new Font("Arial", Font.BOLD, 40));
            FontMetrics metrics = g.getFontMetrics();
            String gameOver = "Game Over";
            g.drawString(gameOver,
                        (LARGHEZZA - metrics.stringWidth(gameOver)) / 2,
                        ALTEZZA / 2 - 40);

            g.setColor(Color.WHITE);
            g.setFont(new Font("Arial", Font.BOLD, 20));
            metrics = g.getFontMetrics();
            String score = "Punteggio: " + punteggio;
```

```
            g.drawString(score,
                         (LARGHEZZA - metrics.stringWidth(score)) / 2,
                         ALTEZZA / 2);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new GiocoSnake());
    }
}
```

Esercizio 7: Gestione Studenti

Un'applicazione per la gestione degli studenti di una classe.

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

public class GestioneStudenti extends JFrame {
    private JTextField txtMatricola, txtNome, txtCognome, txtVoto;
    private JTable tabStudenti;
    private DefaultTableModel modelTabella;
    private ArrayList<Studente> studenti = new ArrayList<>();

    public GestioneStudenti() {
        setTitle("Gestione Studenti");
        setSize(700, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Panel per inserimento dati
        JPanel panInserimento = new JPanel(new GridLayout(5, 2, 10, 10));

        panInserimento.add(new JLabel("Matricola:"));
        txtMatricola = new JTextField();
        panInserimento.add(txtMatricola);

        panInserimento.add(new JLabel("Nome:"));
        txtNome = new JTextField();
        panInserimento.add(txtNome);

        panInserimento.add(new JLabel("Cognome:"));
        txtCognome = new JTextField();
        panInserimento.add(txtCognome);

        panInserimento.add(new JLabel("Voto:"));
```

```
        txtVoto = new JTextField();
        panInserimento.add(txtVoto);

        JButton btnInserisci = new JButton("Inserisci");
        btnInserisci.addActionListener(e -> inserisciStudente());
        panInserimento.add(btnInserisci);

        JButton btnPulisci = new JButton("Pulisci campi");
        btnPulisci.addActionListener(e -> pulisciCampi());
        panInserimento.add(btnPulisci);

        // Panel per tabella e pulsanti azione
        JPanel panTabella = new JPanel(new BorderLayout());

        // Crea tabella
        String[] colonne = {"Matricola", "Nome", "Cognome", "Voto"};
        modelTabella = new DefaultTableModel(colonne, 0) {
            @Override
            public boolean isCellEditable(int row, int column) {
                return false; // Rende la tabella non modificabile
            }
        };

        tabStudenti = new JTable(modelTabella);
        tabStudenti.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        JScrollPane scrollPane = new JScrollPane(tabStudenti);
        panTabella.add(scrollPane, BorderLayout.CENTER);

        // Panel per pulsanti azione
        JPanel panAzioni = new JPanel();

        JButton btnModifica = new JButton("Modifica");
        btnModifica.addActionListener(e -> modificaStudente());

        JButton btnElimina = new JButton("Elimina");
        btnElimina.addActionListener(e -> eliminaStudente());

        JButton btnMedia = new JButton("Calcola Media");
        btnMedia.addActionListener(e -> calcolaMedia());

        panAzioni.add(btnModifica);
        panAzioni.add(btnElimina);
        panAzioni.add(btnMedia);

        panTabella.add(panAzioni, BorderLayout.SOUTH);

        // Layout generale
        JSplitPane splitPane = new JSplitPane(
            JSplitPane.HORIZONTAL_SPLIT,
            panInserimento,
```

```
            panTabella
        );
        splitPane.setDividerLocation(300);

        add(splitPane);

        // Aggiungi alcuni studenti di esempio
        aggiungiStudentiEsempio();

        setVisible(true);
    }

    private void inserisciStudente() {
        try {
            String matricola = txtMatricola.getText().trim();
            String nome = txtNome.getText().trim();
            String cognome = txtCognome.getText().trim();
            double voto = Double.parseDouble(txtVoto.getText().replace(",",
".","));

            // Validazione
            if (matricola.isEmpty() || nome.isEmpty() || cognome.isEmpty())
{
                JOptionPane.showMessageDialog(this, "Tutti i campi sono
obbligatori", "Errore", JOptionPane.ERROR_MESSAGE);
                return;
            }

            if (voto < 0 || voto > 30) {
                JOptionPane.showMessageDialog(this, "Il voto deve essere
compreso tra 0 e 30", "Errore", JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Controlla se la matricola esiste già
            for (Studente s : studenti) {
                if (s.getMatricola().equals(matricola)) {
                    int risposta = JOptionPane.showConfirmDialog(this,
                                    "Matricola già esistente. Vuoi
aggiornare i dati?",
                                    "Matricola esistente",
                                    JOptionPane.YES_NO_OPTION);
                    if (risposta == JOptionPane.YES_OPTION) {
                        s.setNome(nome);
                        s.setCognome(cognome);
                        s.setVoto(voto);
                        aggiornaTabella();
                        pulisciCampi();
                    }
                    return;
```

```
                }
            }

            // Crea nuovo studente
            Studente nuovoStudente = new Studente(matricola, nome, cognome,
voto);
            studenti.add(nuovoStudente);
            aggiornaTabella();
            pulisciCampi();

        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Formato voto non valido",
"Errore", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void modificaStudente() {
        int riga = tabStudenti.getSelectedRow();
        if (riga >= 0) {
            Studente s = studenti.get(riga);
            txtMatricola.setText(s.getMatricola());
            txtNome.setText(s.getNome());
            txtCognome.setText(s.getCognome());
            txtVoto.setText(String.valueOf(s.getVoto()));

            // Rimuovi lo studente dalla lista (verrà inserito con i nuovi
dati)
            studenti.remove(riga);
            aggiornaTabella();
        } else {
            JOptionPane.showMessageDialog(this, "Seleziona uno studente
dalla tabella", "Nessuna selezione", JOptionPane.WARNING_MESSAGE);
        }
    }

    private void eliminaStudente() {
        int riga = tabStudenti.getSelectedRow();
        if (riga >= 0) {
            int risposta = JOptionPane.showConfirmDialog(this,
                                "Sei sicuro di voler eliminare questo
studente?",
                                "Conferma eliminazione",
                                JOptionPane.YES_NO_OPTION);

            if (risposta == JOptionPane.YES_OPTION) {
                studenti.remove(riga);
                aggiornaTabella();
            }
        } else {
            JOptionPane.showMessageDialog(this, "Seleziona uno studente
```

```
dalla tabella", "Nessuna selezione", JOptionPane.WARNING_MESSAGE);
        }
    }

    private void calcolaMedia() {
        if (studenti.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Nessuno studente presente",
"Attenzione", JOptionPane.WARNING_MESSAGE);
            return;
        }

        double somma = 0;
        for (Studente s : studenti) {
            somma += s.getVoto();
        }

        double media = somma / studenti.size();
        JOptionPane.showMessageDialog(this,
                String.format("Media voti: %.2f", media),
                "Media",
                JOptionPane.INFORMATION_MESSAGE);
    }

    private void pulisciCampi() {
        txtMatricola.setText("");
        txtNome.setText("");
        txtCognome.setText("");
        txtVoto.setText("");
        txtMatricola.requestFocus();
    }

    private void aggiornaTabella() {
        modelTabella.setRowCount(0);
        for (Studente s : studenti) {
            Object[] riga = {s.getMatricola(), s.getNome(), s.getCognome(),
s.getVoto()};
            modelTabella.addRow(riga);
        }
    }

    private void aggiungiStudentiEsempio() {
        studenti.add(new Studente("A001", "Mario", "Rossi", 28.5));
        studenti.add(new Studente("A002", "Luigi", "Verdi", 25.0));
        studenti.add(new Studente("A003", "Giulia", "Bianchi", 30.0));
        aggiornaTabella();
    }

    // Classe interna per rappresentare uno studente
    private class Studente {
        private String matricola;
```

```
        private String nome;
        private String cognome;
        private double voto;

        public Studente(String matricola, String nome, String cognome,
double voto) {
            this.matricola = matricola;
            this.nome = nome;
            this.cognome = cognome;
            this.voto = voto;
        }

        public String getMatricola() { return matricola; }
        public String getNome() { return nome; }
        public String getCognome() { return cognome; }
        public double getVoto() { return voto; }

        public void setNome(String nome) { this.nome = nome; }
        public void setCognome(String cognome) { this.cognome = cognome; }
        public void setVoto(double voto) { this.voto = voto; }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new GestioneStudenti());
    }
}
```

Concetti Avanzati da Esplorare

JOptionPane per Dialog

```
// Messaggi informativi
JOptionPane.showMessageDialog(
    parentComponent,  // null o this per centrare sulla finestra
    "Messaggio da mostrare",
    "Titolo",
    JOptionPane.INFORMATION_MESSAGE
);

// Messaggi di errore
JOptionPane.showMessageDialog(
    parentComponent,
    "Messaggio errore",
    "Errore",
    JOptionPane.ERROR_MESSAGE
);

// Messaggi di conferma (Yes/No)
```

```java
int risposta = JOptionPane.showConfirmDialog(
    parentComponent,
    "Domanda?",
    "Conferma",
    JOptionPane.YES_NO_OPTION
);
if (risposta == JOptionPane.YES_OPTION) {
    // Azione se l'utente ha scelto "Sì"
}

// Input dialog
String input = JOptionPane.showInputDialog(
    parentComponent,
    "Inserisci un valore:"
);
```

Personalizzazione dei Componenti

```java
// Personalizzazione di un pulsante
JButton button = new JButton("Pulsante");
button.setBackground(Color.BLUE);
button.setForeground(Color.WHITE);
button.setFont(new Font("Arial", Font.BOLD, 14));
button.setBorder(BorderFactory.createRaisedBevelBorder());
button.setFocusPainted(false);

// Personalizzazione di un pannello
JPanel panel = new JPanel();
panel.setBorder(BorderFactory.createTitledBorder("Titolo Pannello"));
panel.setBackground(new Color(240, 240, 240));
```

Disegnare nel Pannello

```java
JPanel panelDisegno = new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Converti a Graphics2D per funzionalità avanzate
        Graphics2D g2d = (Graphics2D) g;

        // Migliora la qualità del rendering
        g2d.setRenderingHint(
            RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON
        );
```

```
        // Disegna forme
        g2d.setColor(Color.RED);
        g2d.fillRect(20, 20, 100, 50);  // rettangolo pieno

        g2d.setColor(Color.BLUE);
        g2d.drawRect(150, 20, 100, 50); // rettangolo vuoto

        g2d.setColor(Color.GREEN);
        g2d.fillOval(300, 20, 100, 100); // cerchio/ovale pieno

        // Disegna linee
        g2d.setColor(Color.BLACK);
        g2d.setStroke(new BasicStroke(3)); // Spessore linea
        g2d.drawLine(20, 150, 400, 150);

        // Disegna testo
        g2d.setFont(new Font("Arial", Font.BOLD, 20));
        g2d.drawString("Hello, Swing!", 150, 200);
    }
};
```

Suggerimenti ed Errori Comuni

Suggerimenti

1. Usa sempre `SwingUtilities.invokeLater()` per avviare l'interfaccia grafica.
2. Preferisci `ActionListener` con lambda in Java 8+ per codice più pulito.
3. Aggiungi commenti esplicativi per facilitare la comprensione del codice.
4. Separa la logica dell'interfaccia grafica dalla logica di business.
5. Usa i layout manager appropriati invece di posizionamento assoluto.
6. Testa l'interfaccia su diverse risoluzioni dello schermo.

Errori Comuni

1. Modificare i componenti Swing fuori dall'Event Dispatch Thread.
2. Dimenticare di chiamare `setVisible(true)` alla fine del costruttore.
3. Usare layout manager complessi senza comprenderne il funzionamento.
4. Non gestire correttamente gli eventi della finestra (es. chiusura).
5. Creare interfacce troppo complesse in un unico pannello.
6. Dimenticare di disabilitare i componenti quando necessario.

Esercizi Proposti

1. **Rubrica Telefonica**: Crea un'applicazione per memorizzare i contatti con nome, cognome e numero di telefono.

2. **Todo List**: Implementa un'applicazione per gestire una lista di cose da fare.
3. **Disegno**: Crea un'applicazione che permetta di disegnare forme geometriche su un pannello.
4. **Quiz**: Sviluppa un'applicazione che presenti domande a scelta multipla e calcoli il punteggio.
5. **Lettore di Immagini**: Crea un visualizzatore di immagini con funzionalità di navigazione.
6. **Prenotazione Cinema**: Implementa un sistema di prenotazione posti per un cinema.
7. **Calcolatore di Prestiti**: Crea un'applicazione per calcolare le rate di un prestito.
8. **Timer/Cronometro**: Implementa un timer/cronometro con funzionalità start, stop e reset.