```java
// PasswordGenerator.java
import java.awt.BorderLayout;
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.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.StringSelection;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSlider;
import javax.swing.SwingConstants;
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class PasswordGenerator extends JFrame {

    private PasswordField txtPassword;
    private JButton btnGenera, btnCopia, btnSalva, btnVisualizza;
    private JSlider sliderLunghezza;
    private JLabel lblLunghezza, lblRobustezza;
    private JCheckBox chkMaiuscole, chkMinuscole, chkNumeri, chkSpeciali;
    private JPanel pnlRobustezza;

    private ArrayList<String> passwordSalvate;

    public PasswordGenerator() {
        super("Generatore di Password");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);
```

```
        passwordSalvate = new ArrayList<>();

        initComponenti();
        initPannelli();
        initAscoltatori();

        this.pack();
        this.setVisible(true);
    }

    private void initComponenti() {
        // Campo password
        txtPassword = new PasswordField();

        // Pulsanti
        btnGenera = new JButton("Genera Password");
        btnGenera.setFont(new Font("Sans-Serif", Font.BOLD, 14));
        btnGenera.setBackground(new Color(100, 149, 237)); // Cornflower
blue

        btnCopia = new JButton("Copia");
        btnCopia.setBackground(new Color(144, 238, 144)); // Light green

        btnSalva = new JButton("Salva");
        btnSalva.setBackground(new Color(255, 222, 173)); // Navajo white

        btnVisualizza = new JButton("Password Salvate");
        btnVisualizza.setBackground(new Color(216, 191, 216)); // Thistle

        // Slider
        sliderLunghezza = new JSlider(JSlider.HORIZONTAL, 4, 24, 12);
        sliderLunghezza.setMajorTickSpacing(4);
        sliderLunghezza.setMinorTickSpacing(1);
        sliderLunghezza.setPaintTicks(true);
        sliderLunghezza.setPaintLabels(true);

        // Label
        lblLunghezza = new JLabel("Lunghezza: 12");
        lblLunghezza.setFont(new Font("Sans-Serif", Font.BOLD, 14));

        lblRobustezza = new JLabel("Robustezza:");
        lblRobustezza.setFont(new Font("Sans-Serif", Font.BOLD, 14));

        // Checkbox
        chkMaiuscole = new JCheckBox("Lettere maiuscole (A-Z)");
        chkMinuscole = new JCheckBox("Lettere minuscole (a-z)");
        chkNumeri = new JCheckBox("Numeri (0-9)");
        chkSpeciali = new JCheckBox("Caratteri speciali (!@#$%^&*)");
```

```
        // Impostazioni predefinite
        chkMinuscole.setSelected(true);
        chkMaiuscole.setSelected(true);
        chkNumeri.setSelected(true);

        // Pannello robustezza
        pnlRobustezza = new JPanel();
        pnlRobustezza.setPreferredSize(new Dimension(200, 20));
        pnlRobustezza.setBackground(Color.LIGHT_GRAY);

pnlRobustezza.setBorder(BorderFactory.createLineBorder(Color.BLACK));
    }

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

        // Pannello Nord - Campo password
        JPanel pnlNord = new JPanel(new BorderLayout(5, 5));
        pnlNord.setBorder(BorderFactory.createEmptyBorder(10, 10, 5, 10));
        pnlNord.add(txtPassword, BorderLayout.CENTER);

        JPanel pnlPulsanti = new JPanel(new FlowLayout(FlowLayout.CENTER,
10, 0));
        pnlPulsanti.add(btnCopia);
        pnlPulsanti.add(btnSalva);

        pnlNord.add(pnlPulsanti, BorderLayout.EAST);

        // Pannello Centro - Opzioni
        JPanel pnlCentro = new JPanel();
        pnlCentro.setLayout(new BoxLayout(pnlCentro, BoxLayout.Y_AXIS));
        pnlCentro.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));

        // Pannello lunghezza
        JPanel pnlLunghezza = new JPanel(new BorderLayout(5, 5));
        pnlLunghezza.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(),
                "Lunghezza Password",
                TitledBorder.LEFT,
                TitledBorder.TOP));

        pnlLunghezza.add(lblLunghezza, BorderLayout.NORTH);
        pnlLunghezza.add(sliderLunghezza, BorderLayout.CENTER);

        // Pannello opzioni caratteri
        JPanel pnlOpzioni = new JPanel(new GridLayout(4, 1, 0, 5));
        pnlOpzioni.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(),
                "Opzioni Caratteri",
```

```
                TitledBorder.LEFT,
                TitledBorder.TOP));

        pnlOpzioni.add(chkMaiuscole);
        pnlOpzioni.add(chkMinuscole);
        pnlOpzioni.add(chkNumeri);
        pnlOpzioni.add(chkSpeciali);

        // Pannello robustezza
        JPanel pnlRobustezzaContainer = new JPanel(new BorderLayout(5, 5));
        pnlRobustezzaContainer.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(),
                "Valutazione Robustezza",
                TitledBorder.LEFT,
                TitledBorder.TOP));

        pnlRobustezzaContainer.add(lblRobustezza, BorderLayout.NORTH);
        pnlRobustezzaContainer.add(pnlRobustezza, BorderLayout.CENTER);

        // Aggiunta pannelli a pnlCentro
        pnlCentro.add(pnlLunghezza);
        pnlCentro.add(Box.createRigidArea(new Dimension(0, 10)));
        pnlCentro.add(pnlOpzioni);
        pnlCentro.add(Box.createRigidArea(new Dimension(0, 10)));
        pnlCentro.add(pnlRobustezzaContainer);

        // Pannello Sud - Pulsanti
        JPanel pnlSud = new JPanel(new GridLayout(1, 2, 10, 0));
        pnlSud.setBorder(BorderFactory.createEmptyBorder(5, 10, 10, 10));

        pnlSud.add(btnGenera);
        pnlSud.add(btnVisualizza);

        // Aggiunta pannelli al ContentPane
        contentPane.add(pnlNord, BorderLayout.NORTH);
        contentPane.add(pnlCentro, BorderLayout.CENTER);
        contentPane.add(pnlSud, BorderLayout.SOUTH);
    }

    private void initAscoltatori() {
        // Ascoltatore 1: Classe interna per generare password
        btnGenera.addActionListener(new AscoltaGenera());

        // Ascoltatore 2: Classe anonima per copia
        btnCopia.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String password = txtPassword.getText();
                if (!password.isEmpty()) {
                    StringSelection selection = new
```

```
StringSelection(password);
                    Clipboard clipboard =
Toolkit.getDefaultToolkit().getSystemClipboard();
                    clipboard.setContents(selection, selection);
                    JOptionPane.showMessageDialog(PasswordGenerator.this,
                            "Password copiata negli appunti!",
                            "Copia",
                            JOptionPane.INFORMATION_MESSAGE);
                }
            }
        });

        // Ascoltatore 3: Classe esterna per gestione password salvate
        GestionePasswordListener gestorePassword = new
GestionePasswordListener(this);
        btnSalva.addActionListener(gestorePassword);
        btnVisualizza.addActionListener(gestorePassword);

        // Ascoltatore per lo slider
        sliderLunghezza.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                int valore = sliderLunghezza.getValue();
                lblLunghezza.setText("Lunghezza: " + valore);
            }
        });
    }

    // Metodi per gestire le password salvate
    public void aggiungiPassword(String password) {
        passwordSalvate.add(password);
    }

    public ArrayList<String> getPasswordSalvate() {
        return passwordSalvate;
    }

    // Metodo per valutare la robustezza della password
    private void valutaRobustezza(String password) {
        int punteggio = 0;
        int lunghezza = password.length();

        // Valutazione lunghezza
        if (lunghezza >= 8) punteggio += 1;
        if (lunghezza >= 12) punteggio += 1;
        if (lunghezza >= 16) punteggio += 1;

        // Valutazione complessità
        boolean hasMaiuscole = password.matches(".*[A-Z].*");
        boolean hasMinuscole = password.matches(".*[a-z].*");
```

```
        boolean hasNumeri = password.matches(".*\\d.*");
        boolean hasSpeciali = password.matches(".*[!@#$%^&*()_+\\-=\\[\\]
{};':\"\\\\|,.<>/?].*");

        if (hasMaiuscole) punteggio += 1;
        if (hasMinuscole) punteggio += 1;
        if (hasNumeri) punteggio += 1;
        if (hasSpeciali) punteggio += 1;

        // Aggiornamento UI
        Color colore;
        String livello;

        if (punteggio <= 2) {
            colore = Color.RED;
            livello = "Debole";
        } else if (punteggio <= 4) {
            colore = Color.ORANGE;
            livello = "Moderata";
        } else if (punteggio <= 6) {
            colore = Color.YELLOW;
            livello = "Buona";
        } else {
            colore = Color.GREEN;
            livello = "Forte";
        }

        pnlRobustezza.setBackground(colore);
        lblRobustezza.setText("Robustezza: " + livello);
    }

    // Classe interna per generare password
    private class AscoltaGenera implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Verifica se almeno un'opzione è selezionata
            if (!chkMaiuscole.isSelected() && !chkMinuscole.isSelected() &&
                !chkNumeri.isSelected() && !chkSpeciali.isSelected()) {
                JOptionPane.showMessageDialog(PasswordGenerator.this,
                        "Seleziona almeno un tipo di carattere!",
                        "Errore",
                        JOptionPane.ERROR_MESSAGE);
                return;
            }

            int lunghezza = sliderLunghezza.getValue();
            String password = generaPassword(lunghezza);
            txtPassword.setText(password);

            // Valuta robustezza
```

```
            valutaRobustezza(password);
        }

        private String generaPassword(int lunghezza) {
            StringBuilder password = new StringBuilder();
            StringBuilder caratteriPossibili = new StringBuilder();

            // Crea la stringa di caratteri possibili in base alle opzioni
selezionate
            if (chkMaiuscole.isSelected()) {
                caratteriPossibili.append("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
            }

            if (chkMinuscole.isSelected()) {
                caratteriPossibili.append("abcdefghijklmnopqrstuvwxyz");
            }

            if (chkNumeri.isSelected()) {
                caratteriPossibili.append("0123456789");
            }

            if (chkSpeciali.isSelected()) {
                caratteriPossibili.append("!@#$%^&*()_+-=[]{}|;:,.<>?");
            }

            Random random = new Random();
            String caratteri = caratteriPossibili.toString();

            // Assicurati che la password contenga almeno un carattere di
ogni tipo selezionato
            if (chkMaiuscole.isSelected()) {

password.append("ABCDEFGHIJKLMNOPQRSTUVWXYZ".charAt(random.nextInt(26)));
            }

            if (chkMinuscole.isSelected()) {

password.append("abcdefghijklmnopqrstuvwxyz".charAt(random.nextInt(26)));
            }

            if (chkNumeri.isSelected()) {
                password.append("0123456789".charAt(random.nextInt(10)));
            }

            if (chkSpeciali.isSelected()) {
                String speciali = "!@#$%^&*()_+-=[]{}|;:,.<>?";

password.append(speciali.charAt(random.nextInt(speciali.length())));
            }
```

```
            // Genera il resto della password casualmente
            while (password.length() < lunghezza) {
                int index = random.nextInt(caratteri.length());
                password.append(caratteri.charAt(index));
            }

            // Mescola la password
            char[] passwordArray = password.toString().toCharArray();
            for (int i = 0; i < passwordArray.length; i++) {
                int j = random.nextInt(passwordArray.length);
                char temp = passwordArray[i];
                passwordArray[i] = passwordArray[j];
                passwordArray[j] = temp;
            }

            return new String(passwordArray).substring(0, lunghezza);
        }
    }

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

// PasswordField.java
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import javax.swing.BorderFactory;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class PasswordField extends JTextField {

    public PasswordField() {
        super();

        // Impostazioni grafiche
        setFont(new Font("Monospaced", Font.BOLD, 20));
        setHorizontalAlignment(SwingConstants.CENTER);
        setEditable(false);
        setBackground(new Color(245, 245, 245));
        setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createLineBorder(Color.BLACK, 2),
                BorderFactory.createEmptyBorder(10, 10, 10, 10)));

        // Dimensioni
        setPreferredSize(new Dimension(0, 50));
    }

    @Override
```

```
    public void setText(String text) {
        super.setText(text);

        // Se testo presente, cambia sfondo
        if (!text.isEmpty()) {
            setBackground(new Color(240, 248, 255)); // Alice blue
        } else {
            setBackground(new Color(245, 245, 245));
        }
    }
}

// GestionePasswordListener.java
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ListSelectionModel;

public class GestionePasswordListener implements ActionListener {

    private PasswordGenerator app;

    public GestionePasswordListener(PasswordGenerator app) {
        this.app = app;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String comando = e.getActionCommand();

        if (comando.equals("Salva")) {
            // Salva la password corrente
            String password = app.getContentPane().getComponent(0)
                    .getComponent(0).getComponent(0).getName();

            if (password != null && !password.isEmpty()) {
                app.aggiungiPassword(password);
                JOptionPane.showMessageDialog(app,
                        "Password salvata con successo!",
                        "Salvata",
                        JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(app,
```

```
                    "Genera prima una password!",
                    "Errore",
                    JOptionPane.ERROR_MESSAGE);
            }
        } else if (comando.equals("Password Salvate")) {
            // Visualizza le password salvate
            mostraPasswordSalvate();
        }
    }

    private void mostraPasswordSalvate() {
        // Crea un JDialog per visualizzare le password salvate
        JDialog dialog = new JDialog(app, "Password Salvate", true);
        dialog.setLayout(new BorderLayout(10, 10));
        dialog.setSize(400, 300);
        dialog.setLocationRelativeTo(app);

        // Crea una lista per visualizzare le password
        DefaultListModel<String> listModel = new DefaultListModel<>();
        for (String password : app.getPasswordSalvate()) {
            listModel.addElement(password);
        }

        JList<String> listPasswords = new JList<>(listModel);
        listPasswords.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

        JScrollPane scrollPane = new JScrollPane(listPasswords);

        // Pulsante per chiudere il dialog
        JButton btnChiudi = new JButton("Chiudi");
        btnChiudi.addActionListener(e -> dialog.dispose());

        // Pulsante per eliminare una password
        JButton btnElimina = new JButton("Elimina");
        btnElimina.addActionListener(e -> {
            int selectedIndex = listPasswords.getSelectedIndex();
            if (selectedIndex != -1) {
                app.getPasswordSalvate().remove(selectedIndex);
                listModel.remove(selectedIndex);
            }
        });

        // Pulsante per copiare una password
        JButton btnCopia = new JButton("Copia");
        btnCopia.addActionListener(e -> {
            int selectedIndex = listPasswords.getSelectedIndex();
            if (selectedIndex != -1) {
                String password = listPasswords.getSelectedValue();
                java.awt.datatransfer.StringSelection selection =
                        new java.awt.datatransfer.StringSelection(password);
```

```
                java.awt.datatransfer.Clipboard clipboard =

java.awt.Toolkit.getDefaultToolkit().getSystemClipboard();
                clipboard.setContents(selection, selection);
                JOptionPane.showMessageDialog(dialog,
                        "Password copiata negli appunti!",
                        "Copia",
                        JOptionPane.INFORMATION_MESSAGE);
            }
        });

        // Pannello per i pulsanti
        JPanel pnlBottoni = new JPanel();
        pnlBottoni.add(btnCopia);
        pnlBottoni.add(btnElimina);
        pnlBottoni.add(btnChiudi);

        // Aggiunge componenti al dialog
        dialog.add(scrollPane, BorderLayout.CENTER);
        dialog.add(pnlBottoni, BorderLayout.SOUTH);

        // Visualizza il dialog
        dialog.setVisible(true);
    }
}

}
```