```java
// AgendaApp.java
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;

public class AgendaApp extends JFrame {

    private JLabel lblData;
    private JPanel pnlSlots;
    private JButton btnPrecedente, btnSuccessivo, btnOggi,
btnNuovoAppuntamento, btnCerca;
    private JTextField txtCerca;

    private LocalDate dataCorrente;
    private Map<LocalDate, ArrayList<Appuntamento>> appuntamenti;
    private SlotAppuntamento[] slots;

    private final int ORA_INIZIO = 8;
    private final int ORA_FINE = 20;

    public AgendaApp() {
        super("Agenda Appuntamenti");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(800, 600);
        this.setLocationRelativeTo(null);
```

```
        dataCorrente = LocalDate.now();
        appuntamenti = new HashMap<>();

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

        aggiornaVista();

        setVisible(true);
        caricaAppuntamenti();
    }

    private void initComponenti() {
        // Label per la data
        lblData = new JLabel();
        lblData.setHorizontalAlignment(JLabel.CENTER);

        // Pannello per gli slot orari
        pnlSlots = new JPanel();

        // Bottoni di navigazione
        btnPrecedente = new JButton("◀ Precedente");
        btnSuccessivo = new JButton("Successivo ▶");
        btnOggi = new JButton("Oggi");

        // Componenti per nuovo appuntamento e ricerca
        btnNuovoAppuntamento = new JButton("Nuovo Appuntamento");
        btnCerca = new JButton("Cerca");
        txtCerca = new JTextField(15);

        // Inizializzazione slot orari
        slots = new SlotAppuntamento[ORA_FINE - ORA_INIZIO];
        for (int i = 0; i < slots.length; i++) {
            slots[i] = new SlotAppuntamento(ORA_INIZIO + i);
        }
    }

    private void initPannelli() {
        Container contenitore = this.getContentPane();

        // Pannello superiore con data e navigazione
        JPanel pnlNord = new JPanel(new BorderLayout());
        JPanel pnlNavigazione = new JPanel(new FlowLayout());
        pnlNavigazione.add(btnPrecedente);
        pnlNavigazione.add(btnOggi);
        pnlNavigazione.add(btnSuccessivo);
        pnlNord.add(lblData, BorderLayout.CENTER);
        pnlNord.add(pnlNavigazione, BorderLayout.SOUTH);
```

```
        // Pannello centrale con gli slot orari
        pnlSlots.setLayout(new GridLayout(slots.length, 1, 0, 2));
        for (SlotAppuntamento slot : slots) {
            pnlSlots.add(slot);
        }
        JScrollPane scrollPane = new JScrollPane(pnlSlots);

scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS)
;

        // Pannello inferiore con bottoni e ricerca
        JPanel pnlSud = new JPanel(new FlowLayout());
        pnlSud.add(btnNuovoAppuntamento);
        pnlSud.add(new JLabel("Cerca:"));
        pnlSud.add(txtCerca);
        pnlSud.add(btnCerca);

        // Aggiunta pannelli al contenitore principale
        contenitore.add(pnlNord, BorderLayout.NORTH);
        contenitore.add(scrollPane, BorderLayout.CENTER);
        contenitore.add(pnlSud, BorderLayout.SOUTH);
    }

    private void initAscoltatori() {
        // Ascoltatore con classe anonima per i bottoni di navigazione
        btnPrecedente.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dataCorrente = dataCorrente.minusDays(1);
                aggiornaVista();
            }
        });

        btnSuccessivo.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dataCorrente = dataCorrente.plusDays(1);
                aggiornaVista();
            }
        });

        btnOggi.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dataCorrente = LocalDate.now();
                aggiornaVista();
            }
        });

        // Ascoltatore con classe interna per gli slot orari
```

```
        GestoreSlot gestoreSlot = new GestoreSlot();
        for (SlotAppuntamento slot : slots) {
            slot.addMouseListener(gestoreSlot);
        }

        // Ascoltatore con classe esterna per il dialogo e la ricerca
        AscoltaAgenda ascoltaAgenda = new AscoltaAgenda(this);
        btnNuovoAppuntamento.addActionListener(ascoltaAgenda);
        btnCerca.addActionListener(ascoltaAgenda);
    }

    // Metodo per aggiornare la vista con la data corrente
    public void aggiornaVista() {
        // Formato della data
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE d
MMMM yyyy");
        lblData.setText(dataCorrente.format(formatter));

        // Aggiorna gli slot
        for (SlotAppuntamento slot : slots) {
            slot.setAppuntamento(null);
        }

        // Popolamento degli slot con gli appuntamenti della data corrente
        if (appuntamenti.containsKey(dataCorrente)) {
            for (Appuntamento app : appuntamenti.get(dataCorrente)) {
                int ora = app.getOra();
                if (ora >= ORA_INIZIO && ora < ORA_FINE) {
                    slots[ora - ORA_INIZIO].setAppuntamento(app);
                }
            }
        }

        pnlSlots.revalidate();
        pnlSlots.repaint();
    }

    // Metodo per aggiungere un appuntamento
    public void aggiungiAppuntamento(Appuntamento app) {
        LocalDate data = app.getData();

        if (!appuntamenti.containsKey(data)) {
            appuntamenti.put(data, new ArrayList<>());
        }

        // Controllo se c'è già un appuntamento nella stessa ora
        boolean sovrapposizione = false;
        for (Appuntamento esistente : appuntamenti.get(data)) {
            if (esistente.getOra() == app.getOra()) {
                sovrapposizione = true;
```

```
                break;
            }
        }

        if (sovrapposizione) {
            JOptionPane.showMessageDialog(this,
                "Esiste già un appuntamento per le ore " + app.getOra() +
":00",
                "Sovrapposizione", JOptionPane.WARNING_MESSAGE);
            return;
        }

        appuntamenti.get(data).add(app);

        // Aggiorna la vista se la data dell'appuntamento è quella corrente
        if (data.equals(dataCorrente)) {
            aggiornaVista();
        }

        salvaAppuntamenti();
    }

    // Metodo per eliminare un appuntamento
    public void eliminaAppuntamento(Appuntamento app) {
        LocalDate data = app.getData();

        if (appuntamenti.containsKey(data)) {
            appuntamenti.get(data).remove(app);
            aggiornaVista();
            salvaAppuntamenti();
        }
    }

    // Metodo per cercare appuntamenti
    public void cercaAppuntamenti(String keyword) {
        StringBuilder risultato = new StringBuilder();
        int count = 0;

        for (Map.Entry<LocalDate, ArrayList<Appuntamento>> entry :
appuntamenti.entrySet()) {
            for (Appuntamento app : entry.getValue()) {
                if
(app.getTitolo().toLowerCase().contains(keyword.toLowerCase()) ||

app.getDescrizione().toLowerCase().contains(keyword.toLowerCase())) {


risultato.append(app.getData().format(DateTimeFormatter.ofPattern("dd/MM/yyy
y")))
                                .append(" - ")
```

```
                            .append(app.getOra()).append(":00 - ")
                            .append(app.getTitolo()).append("\n");
                    count++;
                }
            }
        }

        if (count > 0) {
            JOptionPane.showMessageDialog(this,
                "Risultati trovati:\n" + risultato.toString(),
                "Risultati ricerca", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this,
                "Nessun risultato trovato per: " + keyword,
                "Ricerca", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    // Metodi per salvare e caricare appuntamenti
    private void salvaAppuntamenti() {
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("agenda.dat"))) {
            out.writeObject(appuntamenti);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this,
                "Errore durante il salvataggio degli appuntamenti: " +
e.getMessage(),
                "Errore", JOptionPane.ERROR_MESSAGE);
        }
    }

    @SuppressWarnings("unchecked")
    private void caricaAppuntamenti() {
        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream("agenda.dat"))) {
            appuntamenti = (Map<LocalDate, ArrayList<Appuntamento>>)
in.readObject();
            aggiornaVista();
        } catch (Exception e) {
            // File non esistente o errore di lettura, inizializza una nuova
mappa
            appuntamenti = new HashMap<>();
        }
    }

    // Getter per data corrente
    public LocalDate getDataCorrente() {
        return dataCorrente;
    }
```

```
    // Classe interna per gestire gli eventi degli slot
    private class GestoreSlot implements java.awt.event.MouseListener {

        @Override
        public void mouseClicked(java.awt.event.MouseEvent e) {
            SlotAppuntamento slot = (SlotAppuntamento) e.getSource();

            if (e.getClickCount() == 2) {
                // Doppio click per modificare o eliminare un appuntamento
                if (slot.getAppuntamento() != null) {
                    String[] opzioni = {"Modifica", "Elimina", "Annulla"};
                    int scelta =
JOptionPane.showOptionDialog(AgendaApp.this,
                                    "Cosa vuoi fare con questo
appuntamento?",
                                    "Gestione appuntamento",
                                    JOptionPane.DEFAULT_OPTION,
                                    JOptionPane.QUESTION_MESSAGE,
                                    null, opzioni, opzioni[0]);

                    if (scelta == 0) {
                        // Modifica
                        modificaAppuntamento(slot.getAppuntamento());
                    } else if (scelta == 1) {
                        // Elimina
                        if (JOptionPane.showConfirmDialog(AgendaApp.this,
                                "Sei sicuro di voler eliminare questo
appuntamento?",
                                "Conferma eliminazione",
                                JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION) {
                            eliminaAppuntamento(slot.getAppuntamento());
                        }
                    }
                } else {
                    // Crea un nuovo appuntamento per questo slot
                    DialogoAppuntamento dialogo = new
DialogoAppuntamento(AgendaApp.this, slot.getOra());
                    dialogo.setVisible(true);
                }
            }
        }

        // Modifica di un appuntamento esistente
        private void modificaAppuntamento(Appuntamento app) {
            DialogoAppuntamento dialogo = new
DialogoAppuntamento(AgendaApp.this, app);
            dialogo.setVisible(true);
        }
```

```
        @Override
        public void mousePressed(java.awt.event.MouseEvent e) {}

        @Override
        public void mouseReleased(java.awt.event.MouseEvent e) {}

        @Override
        public void mouseEntered(java.awt.event.MouseEvent e) {
            SlotAppuntamento slot = (SlotAppuntamento) e.getSource();
            slot.setBackground(new Color(230, 230, 250));
        }

        @Override
        public void mouseExited(java.awt.event.MouseEvent e) {
            SlotAppuntamento slot = (SlotAppuntamento) e.getSource();
            slot.resetBackground();
        }
    }

    // Main per avvio applicazione
    public static void main(String[] args) {
        new AgendaApp();
    }
}

// SlotAppuntamento.java
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.io.Serializable;
import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class SlotAppuntamento extends JPanel {

    private int ora;
    private JLabel lblOra;
    private JTextArea txtContenuto;
    private Appuntamento appuntamento;

    public SlotAppuntamento(int ora) {
        super(new BorderLayout());
        this.ora = ora;

        // Inizializzazione componenti
        lblOra = new JLabel(ora + ":00");
        lblOra.setPreferredSize(new Dimension(60, 30));
```

```
        lblOra.setHorizontalAlignment(JLabel.CENTER);
        lblOra.setFont(new Font("Arial", Font.BOLD, 14));

        txtContenuto = new JTextArea();
        txtContenuto.setEditable(false);
        txtContenuto.setLineWrap(true);
        txtContenuto.setWrapStyleWord(true);

        // Layout
        setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));
        setPreferredSize(new Dimension(0, 80));

        add(lblOra, BorderLayout.WEST);
        add(txtContenuto, BorderLayout.CENTER);

        resetBackground();
    }

    // Metodo per impostare l'appuntamento
    public void setAppuntamento(Appuntamento app) {
        this.appuntamento = app;

        if (app != null) {
            txtContenuto.setText(app.getTitolo() + "\n" +
app.getDescrizione());

            // Cambia colore in base alla tipologia
            switch (app.getTipologia()) {
                case "Lavoro":
                    txtContenuto.setBackground(new Color(255, 200, 200));
                    break;
                case "Personale":
                    txtContenuto.setBackground(new Color(200, 255, 200));
                    break;
                case "Salute":
                    txtContenuto.setBackground(new Color(200, 200, 255));
                    break;
                default:
                    txtContenuto.setBackground(new Color(240, 240, 240));
            }
        } else {
            txtContenuto.setText("");
            txtContenuto.setBackground(new Color(250, 250, 250));
        }
    }

    // Getter per l'appuntamento
    public Appuntamento getAppuntamento() {
        return appuntamento;
    }
```

```
    // Getter per l'ora
    public int getOra() {
        return ora;
    }

    // Reset del colore di sfondo
    public void resetBackground() {
        if (appuntamento == null) {
            setBackground(Color.WHITE);
        }
    }
}

// Appuntamento.java
import java.io.Serializable;
import java.time.LocalDate;

public class Appuntamento implements Serializable {

    private LocalDate data;
    private int ora;
    private String titolo;
    private String descrizione;
    private String tipologia;

    public Appuntamento(LocalDate data, int ora, String titolo, String
descrizione, String tipologia) {
        this.data = data;
        this.ora = ora;
        this.titolo = titolo;
        this.descrizione = descrizione;
        this.tipologia = tipologia;
    }

    // Getter e setter
    public LocalDate getData() {
        return data;
    }

    public void setData(LocalDate data) {
        this.data = data;
    }

    public int getOra() {
        return ora;
    }

    public void setOra(int ora) {
        this.ora = ora;
```

```
    }

    public String getTitolo() {
        return titolo;
    }

    public void setTitolo(String titolo) {
        this.titolo = titolo;
    }

    public String getDescrizione() {
        return descrizione;
    }

    public void setDescrizione(String descrizione) {
        this.descrizione = descrizione;
    }

    public String getTipologia() {
        return tipologia;
    }

    public void setTipologia(String tipologia) {
        this.tipologia = tipologia;
    }
}

// DialogoAppuntamento.java
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.LocalDate;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SpinnerNumberModel;

public class DialogoAppuntamento extends JDialog {

    private AgendaApp agenda;
    private Appuntamento appuntamentoModifica;

    private JTextField txtTitolo;
```

```
    private JTextArea txtDescrizione;
    private JSpinner spnOra;
    private JComboBox<String> cmbTipologia;
    private JButton btnSalva, btnAnnulla;

    // Costruttore per nuovo appuntamento
    public DialogoAppuntamento(AgendaApp agenda, int ora) {
        super(agenda, "Nuovo Appuntamento", true);
        this.agenda = agenda;

        initComponenti(ora);
        initLayout();
        initAscoltatori();

        pack();
        setLocationRelativeTo(agenda);
    }

    // Costruttore per modifica appuntamento
    public DialogoAppuntamento(AgendaApp agenda, Appuntamento app) {
        super(agenda, "Modifica Appuntamento", true);
        this.agenda = agenda;
        this.appuntamentoModifica = app;

        initComponenti(app.getOra());
        initLayout();
        initAscoltatori();

        // Popola i campi con i dati dell'appuntamento
        txtTitolo.setText(app.getTitolo());
        txtDescrizione.setText(app.getDescrizione());
        spnOra.setValue(app.getOra());
        cmbTipologia.setSelectedItem(app.getTipologia());

        pack();
        setLocationRelativeTo(agenda);
    }

    private void initComponenti(int ora) {
        txtTitolo = new JTextField(30);

        txtDescrizione = new JTextArea(5, 30);
        txtDescrizione.setLineWrap(true);
        txtDescrizione.setWrapStyleWord(true);

        // Spinner per l'ora (tra le 8 e le 19)
        spnOra = new JSpinner(new SpinnerNumberModel(ora, 8, 19, 1));

        // ComboBox per la tipologia
        String[] tipologie = {"Lavoro", "Personale", "Salute", "Altro"};
```

```
        cmbTipologia = new JComboBox<>(tipologie);

        btnSalva = new JButton("Salva");
        btnAnnulla = new JButton("Annulla");
    }

    private void initLayout() {
        setLayout(new BorderLayout(10, 10));

        // Pannello per i campi
        JPanel pnlCampi = new JPanel(new GridLayout(4, 2, 5, 5));
        pnlCampi.add(new JLabel("Titolo:"));
        pnlCampi.add(txtTitolo);
        pnlCampi.add(new JLabel("Ora:"));
        pnlCampi.add(spnOra);
        pnlCampi.add(new JLabel("Tipologia:"));
        pnlCampi.add(cmbTipologia);
        pnlCampi.add(new JLabel("Descrizione:"));
        pnlCampi.add(txtDescrizione);

        // Pannello per i bottoni
        JPanel pnlBottoni = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        pnlBottoni.add(btnSalva);
        pnlBottoni.add(btnAnnulla);

        add(pnlCampi, BorderLayout.CENTER);
        add(pnlBottoni, BorderLayout.SOUTH);
    }

    private void initAscoltatori() {
        btnSalva.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                salvaDati();
            }
        });

        btnAnnulla.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dispose();
            }
        });
    }

    private void salvaDati() {
        // Validazione
        if (txtTitolo.getText().trim().isEmpty()) {
            javax.swing.JOptionPane.showMessageDialog(this,
                "Il titolo non può essere vuoto",
```

```
                "Errore", javax.swing.JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Creazione appuntamento
        String titolo = txtTitolo.getText().trim();
        String descrizione = txtDescrizione.getText().trim();
        int ora = (Integer) spnOra.getValue();
        String tipologia = (String) cmbTipologia.getSelectedItem();
        LocalDate data = agenda.getDataCorrente();

        if (appuntamentoModifica != null) {
            // In caso di modifica, elimina il vecchio appuntamento
            agenda.eliminaAppuntamento(appuntamentoModifica);
        }

        // Crea nuovo appuntamento
        Appuntamento app = new Appuntamento(data, ora, titolo, descrizione,
tipologia);
        agenda.aggiungiAppuntamento(app);

        dispose();
    }
}

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

public class AscoltaAgenda implements ActionListener {

    private AgendaApp agenda;

    public AscoltaAgenda(AgendaApp agenda) {
        this.agenda = agenda;
    }

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

        switch (comando) {
            case "Nuovo Appuntamento":
                DialogoAppuntamento dialogo = new
DialogoAppuntamento(agenda, 8);
                dialogo.setVisible(true);
                break;
            case "Cerca":
                JTextField textField = (JTextField) ((JPanel)
e.getSource().getParent()).getComponent(2);
```

```
                String keyword = textField.getText().trim();
                if (!keyword.isEmpty()) {
                    agenda.cercaAppuntamenti(keyword);
                } else {
                    javax.swing.JOptionPane.showMessageDialog(agenda,
                        "Inserisci un termine di ricerca",
                        "Ricerca",
javax.swing.JOptionPane.INFORMATION_MESSAGE);
                }
                break;
        }
    }
}
```