CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

DESARROLLO DE INTERFACES

Batería de preguntas tipo test y ejercicios para preparar el examen final.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

Test

1. Se ha desarrollado una aplicación con interfaz gráfica para editar textos y guardarlos en ficheros de texto. ¿Qué componente visual es el más adecuado para mostrar un mensaje de confirmación antes de salir de la aplicación en caso de que se detecte que hay cambios no guardados?
   a) JOptionPane
   b) JFrame
   c) JavaBean
   d) JCheckBox

2. ¿Por qué es tan importante la realimentación en el diseño de interfaces usables?

   a) Para aumentar el contraste visual de los elementos de la aplicación.
   b) Para que el equipo de desarrollo esté informado constantemente de los errores que hay en la aplicación.
   c) Para que el usuario esté permanentemente informado del estado de sus acciones.
   d) Todas las anteriores respuestas son correctas.

3. Respecto de los principios básicos en la utilización de colores en las interfaces de usuario, ¿cuál es el número de colores diferentes que se recomienda utilizar?
   a) Entre 4 i 5 a la misma ventana y no más de 7 en la interfaz total de la aplicación.
   b) Entre 2 i 3 a la misma ventana y no más de 5 en la interfaz total de la aplicación.
   c) Entre 7 i 8 a la misma ventana y no más de 10 en la interfaz total de la aplicación.
   d) No hay límite. Se emplearán tantos como haga falta para captar la atención de los usuarios.

4. Si InsertDialog y UpdateDialog son clases que derivan de JDialog, que ocurrirá cuando se ejecute el código definido al manejador de eventos siguiente?

```
private void mniShowDialogsActionPerformed(java.awt.event.ActionEvent evt) {
    InsertDialog insertDialog = new InsertDialog(this, true);
    insertDialog.setVisible(true);
    UpdateDialog updateDialog = new UpdateDialog(this, true);
    updateDialog.setVisible(true);
    UpdateView();
}
```

   a) Se mostrará un InsertDialog y, cuando se cierre, se mostrará un UpdateDialog.
   b) Se mostrará un InsertDialog y un UpdateDialog al mismo tiempo.
   c) Sólo se mostrará un InsertDialog.
   d) Sólo se mostrará un UpdateDialog.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

5. Si InsertDialog y UpdateDialog son clases que derivan de JDialog, que ocurrirá cuando se ejecute el código definido al manejador de eventos siguiente?

```
private void mniShowDialogActionPerformed(java.awt.event.ActionEvent evt) {
    InsertDialog insertDialog = new InsertDialog(this, false);
    insertDialog.setVisible(true);
    UpdateDialog updateDialog = new UpdateDialog(this, false);
    updateDialog.setVisible(true);
}
```

a) Se mostrará un InsertDialog y, cuando se cierre, se mostrará un UpdateDialog.
b) Se mostrará un InsertDialog y un UpdateDialog al mismo tiempo.
c) Sólo se mostrará un InsertDialog.
d) Sólo se mostrará un UpdateDialog.

6. ¿Qué propiedad de los objetos JRadioButton tenemos que configurar para relacionar un conjunto de JRadioButtons?
a) model
b) containerPanel
c) buttonGroup
d) name

7. ¿Cuál de los siguientes componentes es considerado un contenedor de nivel superior?
a) JScrollPane
b) JDialog
c) JMenu
d) Todas las opciones son correctas

8. Indica con qué característica de los componentes se relaciona la implementación de la clase Serializable.
a) Con la introspección.
b) Con la persistencia.
c) Con la posibilidad del componente de ser modificado.
d) Con la gestión de eventos.

9. ¿Qué propiedad común a todos los componentes visuales tenemos que configurar para cambiar el icono del ratón cuando, por ejemplo, el ratón entra dentro de la zona del componente?
a) cursor
b) pointer
c) mouseOver
d) icon

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

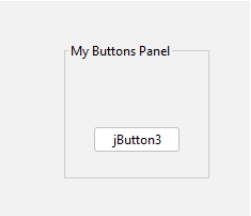
10. ¿Qué componente de la biblioteca swing es comúnmente utilizado para mostrar imágenes en la interfaz?
    a) JImage
    b) JLabel
    c) JMenuItem
    d) JIcon

11. ¿Qué define el comportamiento de un componente JavaBean?
    a) Propiedades.
    b) Métodos y eventos a los que responde.
    c) Su interfaz.
    d) Atributos.

12. ¿Qué propiedad de los objetos JRadioButton tenemos que configurar para relacionar un conjunto de JRadioButtons?
    a) buttonGroup
    b) model
    c) name
    d) containerPanel

13. From which class do every other Swing class derive?
    a) JFrame
    b) JObject
    c) JComponent
    d) JPanel

14. ¿Qué propiedad hay que modificar para añadir a un JPanel un contorno con un texto?
    a) style
    b) outline
    c) margin
    d) border



15. Se ha desarrollado una aplicación con interfaz gráfica para editar textos y guardarlos en ficheros de texto. ¿Qué componente visual es el más adecuado para mostrar un mensaje de confirmación antes de salir de la aplicación en caso de que se detecte que hay cambios no guardados?
    a) JFrame
    b) JavaBean
    c) JOptionPane
    d) JCheckBox

16. Indica con qué característica de los componentes se relaciona la capacidad de un IDE de reconocer los elementos de la interfaz del componente.

a) Con la gestión de eventos.
b) Con la persistencia.
c) Con la posibilidad del componente de ser modificado.
d) Con la introspección.

17. En la aplicación mostrada en la figura, cada vez que se hace click en el botón **bntAddLabel** se añade un JLabel al contenedor **pnlLabels**. ¿Qué layout, de los proporcionados por defecto por Netbeans, es el más adecuado para hacer que los JLabels generados en runtime aparezcan uno debajo del otro?

a) Free design
b) Flow layout
c) Border layout
d) Box layout

18. ¿De qué clase debe heredar un objeto para poder ser utilizado como modelo de un JTable?
a) DefaultTableModel
b) AbstractTableModel
c) JTable
d) ListModel

19. ¿Cuál de los siguientes componentes sería el más adecuado para pedir al usuario que indique su mes de nacimiento?
a) JTextField
b) JLabel
c) JComboBox
d) JList

20. ¿A qué tipo de evento se debe escuchar en un JList para detectar que se ha cambiado el elemento de la lista seleccionado?
a) ItemEvent
b) ValueChangedEvent
c) ActionEvent
d) ListSelectionEvent

21. Desde el punto de vista de la usabilidad de una aplicación, ¿cuántos niveles de diálogos emergentes (modales) conviene no sobrepasar?
a) 1
b) 2
c) 3

d) Mejor no utilizar ningún diálogo modal.

22. ¿Cuál es la salida de la siguiente aplicación?

```
public class MyActionListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent evt) {
        System.out.println("zxc");
    }
}

public class Main extends javax.swing.JFrame implements ActionListener {

    private MyActionListener myX;

    public Main() {
        initComponents();
        setSize(400, 300);
        Timer t = new Timer(400, myX);
        t.start();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("asd");
    }
```

a) Muestra por la consola estándar el mensaje “zxc” cada 0.4 segundos.
b) Muestra por la consola estándar el mensaje “asd” cada 0.4 segundos.
c) Muestra por la consola estándar el mensaje “zxc\nasd” cada 0.4 segundos.
d) No muestra ningún mensaje en la consola.

23. En Java Swing, quin mètode d'un JFrame utilitzaries per assegurar-te que la finestra s’elimini de la memòria quan l'usuari la tanqui, evitant així problemes de fugues de memòria?
a. setVisible(false)
b. dispose()
c. setDefaultCloseOperation(EXIT_ON_CLOSE)
d. pack()

24. En l'ús d’un AbstractTableModel, quin mètode has d'implementar per assegurar-te que els valors de cada cel·la es poden modificar i guardar?
a. getColumnCount()
b. getRowCount()
c. isCellEditable(int rowIndex, int columnIndex)
d. setValueAt(Object aValue, int rowIndex, int columnIndex)

25. Quan afegeixes un ActionListener a un botó en Swing, quin mètode s’executa quan

l'usuari fa clic al botó?
a. onClick()
b. actionPerformed(ActionEvent e)
c. actionTriggered(ActionEvent e)
d. onAction(ActionEvent e)

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

USABILITY AND UX

1. Fill in the gaps with the correct word.

navigation location performance organization familiar
intelligibility visual aesthetics updated visual
readability requirements content layout consistency

a) For the usability of the interface, we must ensure both _____ and _____ of any of the texts it contains.
b) It is very important to keep the consistence of the _____ of the elements of the interface like icons across the different parts or screens of a GUI.
c) Use all available resources to better understand users' _____.
d) Style guides are created to ensure _____ across a product.
e) Ensure that the user interface format meets user expectations, especially related to _____, _____, and _____.
f) The use of _____ formatting and navigation schemes makes it easier for users to learn and remember the _____ of a site.
g) Focus on achieving a high rate of user _____ before dealing with _____.
h) A user interface has to be _____, _____, educational and _____ for the users to feel comfortable, easily access the contents, and make an effective interaction with all the elements of an application.

2. Give 3 examples of how can we give feedback in a graphical user interface.

3. Give 3 examples of restrictions that we can use in graphical user interfaces to reduce the possibility of mistakes.

4. Which are the two interaction gulfs? Explain them.

5. What option would be better for an icon that wants to represent an object?
a) A black and white children's hand-drawn picture of the object.
b) A plain and without many details schematic of the object.
c) A photography of the object.
d) A 3D model of the object.

6. Mark as true (T) or false (F).
- For good readability, line spacing between 1 and 4 points higher than the dimension of the font should be used. True False
- The human eye can detect 16 colours “at a glance”. True False

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

- The optimal contrast level between the font colour and background colour must be of 80% or larger. True False
- Warm colours exhibit calmness and peace (best for office use), while cool colours convey energy and joy (best for personal messages). True False

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

Exercicis

1. En la aplicación mostrada en la figura, cada vez que se hace click en el botón **bntAddLabel** se añade un JLabel al contenedor **pnlLabels**. Escribe el código necesario que tendría que haber en el manejador de eventos del btnAddLabel para implementar la funcionalidad descrita. Supón que hay una variable de clase **int count = 0;** declarada e inicializada.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

2. En el ejercicio anterior, queremos que el texto de los labels cambie a color rojo cuando el ratón 'se pone encima' de un label, y vuelva a tener color negro cuando el ratón 'sale' del área del label. Escribe el código que se tiene que añadir al anterior para dotar de esta funcionalidad a la aplicación.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

3. La siguiente aplicación hace una cuenta atrás y muestra un mensaje al acabar, pero no está funcionando. Explica por qué motivo/s no funciona y que se tiene que hacer para solucionarlo.

```java
package temporizador;

public class TestTemporizador {

    public static void main(String args[]) {
        TemporizadorBean temporizador = new TemporizadorBean();
        temporizador.setActivo(true);
    }
}
```

```java
package temporizador;

import java.io.Serializable;
import javax.swing.Timer;
import java.awt.event.ActionEvent;

public class TemporizadorBean implements Serializable {
    private int tiempo;
    private final Timer t;

    public TemporizadorBean() {
        tiempo = 5;
        t = new Timer(1000, this);
        setActivo(true);
    }

    public final void setActivo(boolean valor) {
        if(valor == true) {
            t.start();
        }
        else {
            t.stop();
        }
    }

    public boolean getActivo() {
        return t.isRunning();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        tiempo--;
        if (tiempo == 0) {
            setActivo(false);
            System.out.println("Terminado");
        }
    }
}
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

4. Se desea crear una aplicación para animar un JLabel, de forma que se desplace a la derecha a una velocidad de 1 pixel por cada 10 ms. El JLabel se empezará a mover cuando se pulse el botón ***Animar*** y se parará cuando se vuelva a pulsar el botón Animar o cuando se haya desplazado 200 píxeles.
A partir del código siguiente, se pide:

a) (2p) Indicar las modificaciones a hacer y escribir el código necesario en el método adecuado para desplazar el JLabel 1 píxel cada 10 ms cumpliendo las condiciones indicadas en el enunciado.

b) (2p) Escribir el código necesario para añadir un listener al botón de forma que al hacer clic en él se invoque el método ***btnAnimateActionPerformed***. Escribir el código que debería haber dentro de *btnAnimateActionPerformed*.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

Main.java

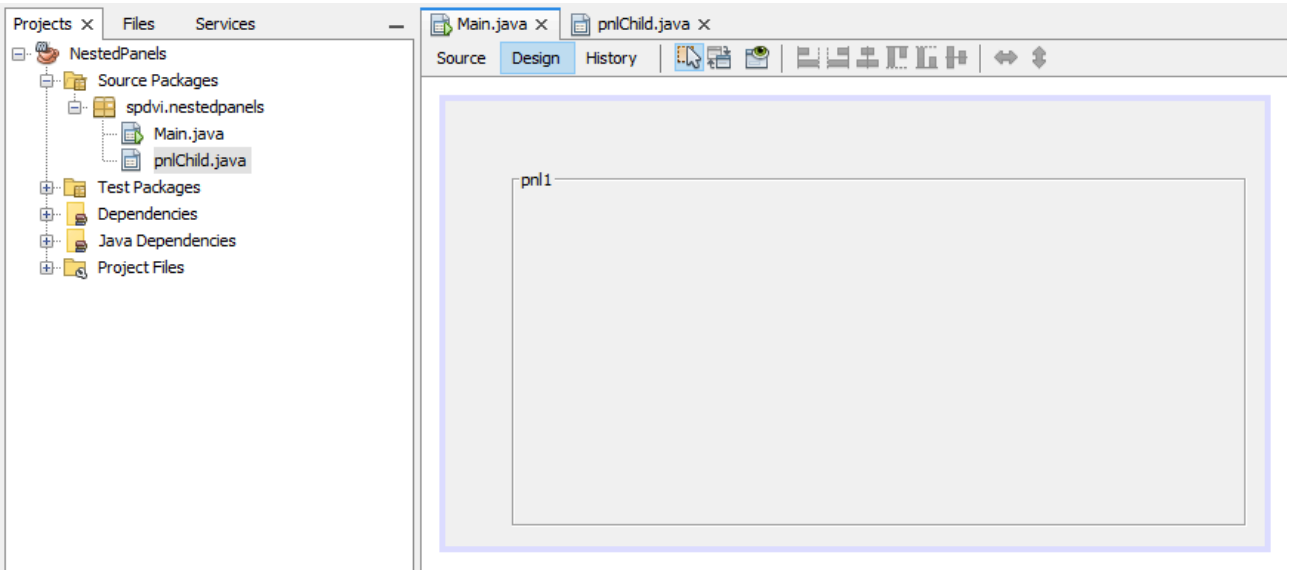
```java
package spdvi.animation;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Main extends JFrame {

    JLabel jLabel1;
    JButton btnAnimate;
    Timer timer;
    int xPosition = 20, yPosition = 100;

    public Main() {
        this.setLocationRelativeTo(null);
        this.setLayout(null);
        this.setPreferredSize(new Dimension(400, 200));
        this.setDefaultCloseOperation(
            javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1 = new JLabel("jLabel1");
        jLabel1.setBounds(xPosition, yPosition, 50, 50);
        jLabel1.setBorder(BorderFactory.createLineBorder(
            new java.awt.Color(255, 0, 51)));

        btnAnimate = new JButton("Animate");
        btnAnimate.setBounds(20, 20, 120, 22);

        this.add(jLabel1);
        this.add(btnAnimate);

        this.pack();
    }

    public void btnAnimateActionPerformed(ActionEvent evt) {

    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Main().setVisible(true);
            }
        });
    }

}
```

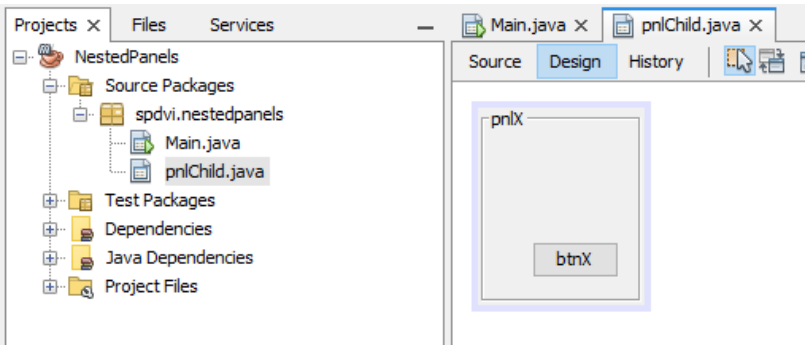
CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

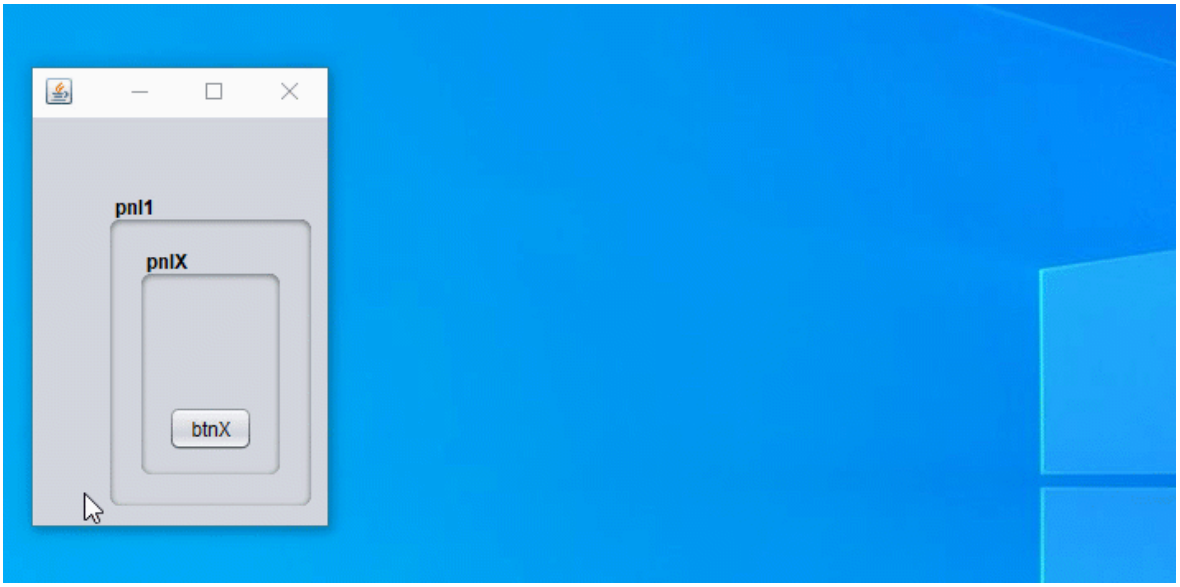
5. Describe un fallo de usabilidad que detectaste en la práctica de la unidad 4 y explica cómo lo solucionaste.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

6. En l'aplicació següent, escriu la instrucció o instruccions que s'ha d'afegir al manejador d'events del component btnX per aconseguir el comportament mostrat en la animació.







```
private void btnXActionPerformed(java.awt.event.ActionEvent evt) {




}
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

7. Per quina línia de codi hem de sustituir la línia comentada perquè TemporizadorBean faci un compte enrerre de 5 segons quan sigui activat?

```
public class TemporizadorBean extends JLabel implements Serializable, ActionListener {
    private int tiempo;
    private final Timer t;

    public TemporizadorBean() {
        tiempo = 5;
        // TODO: What line here?
        setText(Integer.toString(tiempo));
        setActivo(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        setText(Integer.toString(tiempo));
        repaint();
        tiempo--;
        if (tiempo == 0) {
            setActivo(false);
            JOptionPane.showMessageDialog(null, "Terminado", "Aviso", JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

8. Escribe una instrucción para situar un JLabel myLabel en la posición x = 100, y = 40 de su objeto contenedor (suponiendo que tiene layout null), y hacer que tenga un tamaño de 90x16 px.

9. Write a line of code to set the layout of the main window of a Java Swing application to null.

```
public class Main extends javax.swing.JFrame {

        public Main() {


        }

}
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

11. Escribe la instrucción que debería aparecer en la línea 21 para que el mediaPlayerComponent se mostrara dentro del pnlVideo ocupando todo el espacio disponible de éste.

```
5   public class Main extends javax.swing.JFrame {
6
7       private javax.swing.JPanel pnlVideo;
8       private final EmbeddedMediaPlayerComponent mediaPlayerComponent;
9
10      public Main() {
11          initComponents();
12
13          pnlVideo = new javax.swing.JPanel();
14          mediaPlayerComponent = new EmbeddedMediaPlayerComponent();
15
16          getContentPane().setLayout(mgr: null);
17          setSize(width: 1024, height: 768);
18
19          pnlVideo.setBorder(border: javax.swing.BorderFactory.createTitledBorder(title: "Video player"));
20          pnlVideo.setLayout(new java.awt.BorderLayout());
21          /* TODO: Which line here? */
22      }
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

12. Completa la clase **MyJPanel** como corresponda para que, si se hace un **swipe** right sobre un objeto de tipo MyJPanel su color de fondo cambie a Color.ORANGE y si se hace un swipe left el color de fondo sea Color.PINK. La sensibilidad del swipe deberá ser de 10 pixels, es decir, para que se considere que se ha hecho swipe, el ratón se debe haber desplazado al menos 10 pixels.

```
public class MyJPanel extends javax.swing.JPanel {

    public MyJPanel() {
        initComponents();

    }

    @SuppressWarnings("unchecked")
    Generated Code


    // Variables declaration - do not modify
    // End of variables declaration
}
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

13. **Completa el codi de la finestra** (2 punts)
Observa el següent codi incomplet que crea una finestra amb el títol "Aplicació de Prova" i una mida de 400x200 píxels. Completa'l per afegir un JLabel amb el text "Benvingut!" al centre de la finestra.

```
import javax.swing.*;

public class AplicacioProva {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Aplicació de Prova");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // COMPLETA el codi per afegir el JLabel



        frame.setVisible(true);
    }
}
```

14. Es demana que, en un formulari de Swing amb camps com `JTextField` per a nom i cognoms, i un `JButton` de nom `btnSubmit`, el `JButton` es deshabiliti automàticament si algun dels camps és buit. Quan tots els camps tenen text, el botó hauria d'estar activat. Escriu el codi per implementar aquesta funcionalitat.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

15. Implementació d'un Filtre en un JList utilitzant DAO (3 punts)
En aquest exercici, has de completar el codi per implementar un filtre dinàmic. La interfície carrega una llista d'usuaris utilitzant un objecte UserDAO (patró DAO) i mostra aquests usuaris en un JList. A més, hi ha un JTextField que ha de permetre filtrar els usuaris en temps real a mesura que s'escriu en el camp de text.
Esquema del DAO: La classe UserDAO conté un mètode getAllUsers() que retorna una llista de noms d'usuaris (List<String>).
Completa el codi per implementar la funcionalitat de filtre:

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

public class UserFilterApp {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Filtre d'Usuaris");
        JTextField filterField = new JTextField(15);
        JList<String> userList = new JList<>();

        frame.setLayout(new FlowLayout());
        frame.add(new JLabel("Filtre d'usuaris:"));
        frame.add(filterField);
        frame.add(new JScrollPane(userList));

        // Crea un objecte DAO per carregar usuaris
        UserDAO userDao = new UserDAO();
        List<String> allUsers = userDao.getAllUsers();

        // Inicialitza el `JList` amb tots els usuaris
        userList.setListData(allUsers.toArray(new String[0]));

        // Afegeix un DocumentListener per actualitzar el filtre cada vegada que es canvia el
text
        filterField.getDocument().addDocumentListener(new DocumentListener() {
            public void insertUpdate(DocumentEvent e) {
                filterUsers();
            }
            public void removeUpdate(DocumentEvent e) {
                filterUsers();
            }
            public void changedUpdate(DocumentEvent e) {
                filterUsers();
            }
// COMPLETA aquest mètode per implementar el filtre
            private void filterUsers() {

            }
        });

        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

Pregunta 3: Completa el formulari mestre-detall (6 punts)
Afegim codi dins del ActionListener perquè el text introduït en nomField es mostri en resultatLabel.

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

public class FormulariMestreDetall {
    // Mapa que emmagatzema els projectes amb les seves tasques
    private static Map<String, List<String>> projectesTasques = new HashMap<>();

    public static void main(String[] args) {
        JFrame frame = new JFrame("Formulari Mestre-Detall");
        frame.setLayout(new BorderLayout());

        // Panell mestre
        JPanel panellMestre = new JPanel();
        JLabel projecteLabel = new JLabel("Selecciona un projecte:");
        JComboBox<String> projecteComboBox = new JComboBox<>();
        panellMestre.add(projecteLabel);
        panellMestre.add(projecteComboBox);

        // Panell detall
        JPanel panellDetall = new JPanel();
        panellDetall.setLayout(new BoxLayout(panellDetall, BoxLayout.Y_AXIS));
        JLabel tasquesLabel = new JLabel("Tasques:");
        JList<String> tasquesList = new JList<>();
        JScrollPane tasquesScroll = new JScrollPane(tasquesList);
        JTextField tascaField = new JTextField(15);
        JButton afegirButton = new JButton("Afegeix tasca");

        panellDetall.add(tasquesLabel);
        panellDetall.add(tasquesScroll);
        panellDetall.add(new JLabel("Nova tasca:"));
        panellDetall.add(tascaField);
        panellDetall.add(afegirButton);

        frame.add(panellMestre, BorderLayout.NORTH);
        frame.add(panellDetall, BorderLayout.CENTER);
        // Completa la inicialització del JComboBox amb els projectes
        inicialitzarProjectesComboBox(projecteComboBox);
        // Completa la funcionalitat per actualitzar les tasques quan es selecciona un projecte
        projecteComboBox.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String projecteSeleccionat = (String) projecteComboBox.getSelectedItem();
                List<String> tasques = getTasquesDelProjecte(projecteSeleccionat);
                // Implementa la lògica per actualitzar la llista de tasques
```

```
            }
        });
        // Afegir tasca
        afegirButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String tasca = tascaField.getText().trim();
                // Completa la validació de la tasca (el nom de la tasca no pot estar buit i s'ha
d'avisar en cas contrari)
                String projecteSeleccionat = (String) projecteComboBox.getSelectedItem();
                // Completa l'afegit de la tasca a la llista del projecte seleccionat
                afegirTascaAlProjecte(projecteSeleccionat, tasca);
                // Actualitza la llista de tasques
                actualitzarLlistaTasques(projecteSeleccionat);
                tascaField.setText("");
            }
        });
        // Elimina tasca
        tasquesList.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                if (e.getClickCount() == 2) {
                    int index = tasquesList.locationToIndex(e.getPoint());
                    String tascaSeleccionada = tasquesList.getModel().getElementAt(index);
                    String projecteSeleccionat = (String) projecteComboBox.getSelectedItem();
                    // Completa l'eliminació de la tasca seleccionada de la llista
                    eliminarTascaDelProjecte(projecteSeleccionat, tascaSeleccionada);
                    // Actualitza la llista de tasques
                    actualitzarLlistaTasques(projecteSeleccionat);
                }
            }
        });
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    // Completa la inicialització de dades per als projectes
    private static void inicialitzarProjectesComboBox(JComboBox<String> projecteComboBox) {
        // Omple el JComboBox amb les claus del mapa projectesTasques (els noms dels projectes)
    }
    // Completa el mètode per obtenir les tasques d'un projecte
    private static List<String> getTasquesDelProjecte(String projecte) {
        // Retorna la llista de tasques associades a un projecte
        return null;
    }
    // Completa el mètode per afegir una tasca a un projecte
    private static void afegirTascaAlProjecte(String projecte, String tasca) {
        // Afegeix la tasca a la llista de tasques del projecte seleccionat
    }
    // Completa el mètode per eliminar una tasca d'un projecte
    private static void eliminarTascaDelProjecte(String projecte, String tasca) {
        // Elimina la tasca de la llista de tasques del projecte seleccionat
    }
    // Completa el mètode per actualitzar la llista de tasques mostrada
    private static void actualitzarLlistaTasques(String projecte) {
        // Actualitza la llista de tasques mostrada per al projecte seleccionat
    }
    // Mostra un missatge d'error
```

```
    private static void mostrarMissatgeError(String missatge) {
        JOptionPane.showMessageDialog(null, missatge, "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

Instruccions per implementar:

1. **Inicialització del JComboBox:** Has de completar el mètode inicialitzarProjectesComboBox, que ha d'emplenar el JComboBox amb els noms dels projectes.
2. **Obtenir tasques del projecte seleccionat:** El mètode getTasquesDelProjecte ha de retornar la llista de tasques per al projecte seleccionat. L'alumne ha de gestionar la lògica per a obtenir aquesta informació del mapa projectesTasques.
3. **Afegir tasques:** El mètode afegirTascaAlProjecte ha d'afegeir una tasca a la llista del projecte seleccionat en el mapa projectesTasques.
4. **Eliminar tasques:** El mètode eliminarTascaDelProjecte ha de permetre eliminar una tasca seleccionada de la llista del projecte seleccionat.
5. **Actualitzar la llista de tasques:** El mètode actualitzarLlistaTasques ha d'actualitzar el JList perquè es mostrin les tasques actualitzades després d'afegir o eliminar tasques.
6. **Validació i missatges d'error:** El mètode mostrarMissatgeError ja està implementat per a mostrar errors, però l'alumne ha d'implementar correctament la validació a l'afegir una tasca.

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

Exercici 4: Formulari amb Diàlegs Modals i No Modals (4 punts)

En aquest exercici, has de crear una petita aplicació de gestió de tasques que permeti als usuaris afegir tasques a una llista, mostrar missatges informatius mitjançant diàlegs modals i no modals, i confirmar accions com eliminar tasques. El formulari ha de tenir els següents components i funcionalitats:

1. **Afegir Tasca:** L'usuari pot introduir una tasca en un JTextField i afegir-la a una llista. Si la tasca ja existeix, es mostrarà un diàleg modal d'error.
2. **Eliminar Tasca:** L'usuari pot eliminar una tasca fent clic sobre una tasca de la llista. Un diàleg modal de confirmació ha de preguntar si l'usuari està segur abans d'eliminar-la.
3. **Mostrar Tasques:** Les tasques s'han de mostrar en un JList, i quan l'usuari passa el cursor sobre elles, s'ha de mostrar un diàleg no modal amb informació sobre la tasca (per exemple, la data d'afegit).
4. **Diàleg de Missatge:** Quan l'usuari faci una acció important (com afegir una tasca), un diàleg modal ha de mostrar un missatge indicant que l'acció s'ha realitzat correctament.

Teniu les capçaleres dels mètodes al final del codi.

Codi exercici 4

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

public class FormulariAmbDials {
    // Llista de tasques
    private static ArrayList<String> tasques = new ArrayList<>();

    public static void main(String[] args) {
        JFrame frame = new JFrame("Gestió de Tasques");
        frame.setLayout(new BorderLayout());
        initComponents();
        // Panell per afegir tasques
        JPanel panellAfegir = new JPanel();
        JTextField tascaField = new JTextField(15);
        JButton afegirButton = new JButton("Afegeix Tasca");
        panellAfegir.add(tascaField);
        panellAfegir.add(afegirButton);

        // Llista de tasques
        DefaultListModel<String> modelTasques = new DefaultListModel<>();
        JList<String> tasquesList = new JList<>(modelTasques);
        JScrollPane tasquesScroll = new JScrollPane(tasquesList);

        frame.add(panellAfegir, BorderLayout.NORTH);
        frame.add(tasquesScroll, BorderLayout.CENTER);

        // Afegir tasca
        afegirButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String tasca = tascaField.getText().trim();
                if (tasca.isEmpty()) {
                    // Completa la lògica per mostrar un missatge d'error
                } else if (tasques.contains(tasca)) {
```

CIFP Pau Casesnoves
Joan Miró, 22 07300 Inca
Tel. 971 881710 11
secretaria@paucasesnovescifp.cat

```
                    // Completa la lògica per mostrar un missatge d'error
                } else {
                    // Completa la lògica per afegir la tasca
                    // Completa la lògica per mostrar un missatge d'èxit
                }
            }
        });
        // Eliminar tasca amb diàleg modal de confirmació
        tasquesList.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                if (e.getClickCount() == 2) {
                    int index = tasquesList.locationToIndex(e.getPoint());
                    String tascaSeleccionada =
tasquesList.getModel().getElementAt(index);

                    // Completa la lògica per mostrar un diàleg modal de confirmació
                    // Si es confirma, elimina la tasca
                }
            }
        });
        // Diàleg no modal (mostra informació quan el cursor està sobre una tasca)
        tasquesList.addMouseMotionListener(new MouseMotionAdapter() {
            public void mouseMoved(MouseEvent e) {
                int index = tasquesList.locationToIndex(e.getPoint());
                if (index != -1) {
                    String tascaSeleccionada =
tasquesList.getModel().getElementAt(index);
                    // Completa la lògica per mostrar un diàleg no modal amb la tasca
seleccionada
                }
            }
        });
        // Configura finestra
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    // Diàleg modal d'error
    private static void mostrarMissatgeError(String missatge) {
        // Completa la lògica per mostrar un missatge d'error
    }

    // Diàleg modal d'èxit
    private static void mostrarMissatgeExito(String missatge) {
        // Completa la lògica per mostrar un missatge d'èxit
    }

    // Diàleg modal de confirmació
    private static int mostrarDiologoConfirmacio(String missatge) {
        // Completa la lògica per mostrar un diàleg de confirmació i retornar la resposta
        return 0; // Canvia aquest valor per a retornar la resposta correcta
    }

    // Diàleg no modal d'informació
    private static void mostrarDiàlegInformatiu(String tasca) {
        // Completa la lògica per mostrar un diàleg no modal amb informació de la tasca
    }
}
```