Marcos Niño Gutierrez

# Swing_Práctica02_compendio_neRA1

---

# Documento de Usabilidad - Proyecto de Reservas Hotel "El Mirador"

# Documento de Usabilidad - Explicación de Métodos de la Clase `Main`

## 1. Introducción

El código proporcionado corresponde a la clase `Main`, que representa la ventana principal de la aplicación para gestionar las reservas del hotel *El Mirador*. En esta clase se gestionan las interacciones del usuario con los menús, botones y diálogos emergentes, proporcionando una experiencia intuitiva y fluida.

A continuación se explica cómo se utilizan los métodos clave dentro de esta clase, así como la estructura general de la interfaz de usuario.

## 2. Métodos Principales de la Clase `Main`

### 2.1 Método `Main()`

Este es el **constructor de la clase** que inicializa la interfaz gráfica de usuario (GUI) y configura todos los elementos visuales y funcionales de la ventana principal.

- **Configuración de la ventana principal:**

- `setTitle("Gestión Hotel | El Mirador")`: Establece el título de la ventana principal, mostrando el nombre de la aplicación, lo que proporciona claridad al usuario sobre el propósito de la ventana.
- `setSize(800, 400)`: Establece el tamaño de la ventana principal. Este tamaño fue seleccionado para ofrecer una vista cómoda y organizada de los elementos de la interfaz.
- `setLocationRelativeTo(null)`: Centra la ventana en la pantalla, lo que asegura que la ventana se muestre de manera óptima para el usuario.
- `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)`: Establece que la aplicación se cierre completamente cuando el usuario cierre la ventana principal.
- `setIconImage(new ImageIcon(getClass().getResource("/Media/hotel.png")).getImage())`: Configura el ícono de la ventana principal utilizando una imagen relacionada con el hotel. Esto ayuda a los usuarios a reconocer la aplicación visualmente en la barra de tareas.

```java
public Main() {
    // Configuración de la ventana principal
    setTitle("Gestión Hotel | El Mirador");
    setSize(800, 400);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setIconImage(new ImageIcon(getClass().getResource("/Media/hotel.png")).getImage());


    // Configuración de la barra de menú
    JMenuBar menuBar = new JMenuBar();
```

- **Configuración de la barra de menú:**
  - `JMenuBar menuBar = new JMenuBar()` y `setJMenuBar(menuBar)`: Se crea una barra de menú y se asigna a la ventana principal, proporcionando un acceso fácil a diferentes opciones y funcionalidades de la aplicación.
  - **Menú Archivo (`menuArchivo`):** Contiene una opción para salir de la aplicación. La opción de **"Salir"** utiliza un `ActionListener` para ejecutar `System.exit(0)`, lo que termina la ejecución de la aplicación.
  - **Menú Registro (`menuRegistro`):** Contiene dos opciones:
    - **Alta Reservas**: Con un atajo de teclado `Ctrl+A`, permite abrir la ventana de reservas utilizando el método `new AltaReservasDialog(this)`.
    - **Baja Reservas**: Con un atajo de teclado `Ctrl+B`, muestra un mensaje de información diciendo que esta opción no está desarrollada aún, mediante el método `mostrarDialogo`.
  - **Menú Ayuda (`menuAyuda`):** Contiene la opción **"Acerca de..."**, que muestra información sobre el hotel en un cuadro de diálogo, proporcionando al usuario información relevante de manera accesible.

- **Panel central con los botones de "Alta Reservas" y "Baja Reservas":**
    - **`JPanel panelCentral = new JPanel(new GridLayout(1, 2, 10, 10))`:** Se crea un panel central que contiene dos botones, alineados en una fila (GridLayout de 1 fila y 2 columnas). Esto proporciona una estructura ordenada y fácil de entender para el usuario.
    - **Botones de acción:**
        - **`btnAltaReservas`** y **`btnBajaReservas`**: Los botones permiten al usuario acceder a las funciones de **Alta Reservas** y **Baja Reservas**. Cada botón está acompañado de un ícono visual, lo que mejora la accesibilidad y la comprensión de la acción correspondiente. Los botones están configurados con **`ActionListeners`** que ejecutan los métodos correspondientes.
        - **Acción del botón "Alta Reservas"**: Al hacer clic en el botón "Alta Reservas", se abre un nuevo diálogo con el formulario para ingresar los datos de la reserva, utilizando el método `new AltaReservasDialog(this)`.
        - **Acción del botón "Baja Reservas"**: Al hacer clic en el botón "Baja Reservas", se muestra un mensaje indicando que esta opción aún no está disponible, utilizando el método `mostrarDialogo`.

```
// Configuración de la barra de menú
JMenuBar menuBar = new JMenuBar();

// Menú Archivo
JMenu menuArchivo = new JMenu("Archivo");
JMenuItem menuItemSalir = new JMenuItem("Salir");
// Acción para salir de la aplicación
menuItemSalir.addActionListener(e -> System.exit(0));
menuArchivo.add(menuItemSalir);

// Menú Registro
JMenu menuRegistro = new JMenu("Registro");

// Menú de "Alta Reservas" con atajo de teclado Ctrl+A
JMenuItem menuItemAltaReservas = new JMenuItem("Alta Reservas");
menuItemAltaReservas.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A, InputEvent.CTRL_DOWN_MASK));
menuItemAltaReservas.addActionListener(e -> new AltaReservasDialog(this));

// Menú de "Baja Reservas" con atajo de teclado Ctrl+B
JMenuItem menuItemBajaReservas = new JMenuItem("Baja Reservas");
menuItemBajaReservas.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_B, InputEvent.CTRL_DOWN_MASK));
menuItemBajaReservas.addActionListener(e -> mostrarDialogo("Información", "La opción 'Baja Reservas' no está desarrollada aún."));

menuRegistro.add(menuItemAltaReservas);
menuRegistro.add(menuItemBajaReservas);

// Menú Ayuda
JMenu menuAyuda = new JMenu("Ayuda");

// Menú de "Acerca de"
JMenuItem menuItemAcercaDe = new JMenuItem("Acerca de...");
menuItemAcercaDe.addActionListener(e -> mostrarDialogo("Acerca de", "Acerca de Nosotros\r\n"
        + "\r\n"
        + "Bienvenidos a El Mirador, un destino único donde el confort y la excelencia se encuentran en armonía. Ubicado en el corazón de Almería,"
        + "\r\n"
        + "nuestro hotel es el refugio perfecto para aquellos que buscan una experiencia inolvidable,"
        + "\r\n"
        + " ya sea para un viaje de negocios, una escapada romántica o unas vacaciones en familia.\r\n"
        + "\r\n"
        + "Con una decoración moderna y acogedora, [Nombre del Hotel] ofrece habitaciones espaciosas"
        + "\r\n"
        + " y equipadas con todas las comodidades necesarias para hacer de su estancia un verdadero placer."
        + "\r\n"
        + " Nuestros servicios incluyen un restaurante gourmet, un spa de relajación, gimnasio, y acceso a espacios diseñados para eventos especiales."
        + "\r\n"
        + " Además, nuestra ubicación privilegiada le permitirá disfrutar de las principales atracciones turísticas de la zona, así como de un ambiente tranquilo y exclusivo.\r\n"
        + "\r\n"
        + "En El Mirador, nos dedicamos a brindarle el mejor servicio, con un equipo de profesionales comprometidos a hacer de su visita una experiencia única."
        + "\r\n"
        + " Venga y descubra lo que significa sentirse como en casa, mientras disfruta de un servicio de clase mundial y de las mejores vistas.\r\n"
        + "\r\n"
        + "Esperamos darle la bienvenida pronto."));
menuAyuda.add(menuItemAcercaDe);

// Añadir menús a la barra de menús
menuBar.add(menuArchivo);
menuBar.add(menuRegistro);
menuBar.add(menuAyuda);
```

## 2.2 Método `mostrarDialogo(String titulo, String mensaje)`

Este método se utiliza para mostrar mensajes informativos al usuario en forma de un cuadro de diálogo. Es utilizado en dos lugares clave dentro de la clase `Main`:

- En el **menú "Baja Reservas"**, para informar al usuario que esta opción aún no está disponible.
- En el **menú "Acerca de..."**, para mostrar detalles sobre el hotel cuando el usuario selecciona esta opción.

El método usa **`JOptionPane.showMessageDialog`** para mostrar el mensaje en una ventana emergente con el título y el mensaje que se le pase como parámetros. Esto proporciona una forma clara y directa de comunicar al usuario información importante o de advertencia.

```
private void mostrarDialogo(String titulo, String mensaje) {
    JOptionPane.showMessageDialog(this, mensaje, titulo, JOptionPane.INFORMATION_MESSAGE);
}
```

## 2.3 Método `main(String[] args)`

Este es el punto de entrada principal del programa, el cual se ejecuta al iniciar la aplicación. Utiliza **`SwingUtilities.invokeLater()`** para asegurarse de que la creación de la ventana principal se realice en el hilo de eventos de Swing, lo que garantiza que la interfaz gráfica se maneje correctamente y no cause bloqueos o problemas de rendimiento.

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

## Documento de Usabilidad: Alta Reservas Dialog

### 1. Introducción

El componente `AltaReservasDialog` está diseñado para permitir la gestión de reservas en el sistema de un hotel llamado "El Mirador". Este formulario recopila la información del cliente, las fechas de estancia, los detalles de la habitación y los extras solicitados, incluyendo un sistema para calcular el importe de la reserva y un descuento promocional.

**Ventana del diálogo:**

- `AltaReservasDialog(JFrame parent)` **es el constructor de la clase** `AltaReservasDialog`**, que extiende de** `JDialog`**. El** `true` **en el parámetro indica que el diálogo es modal, lo que significa que el usuario debe interactuar con él antes de poder acceder a otras ventanas.**
- `setSize(screenSize.width, screenSize.height)` **establece el tamaño del diálogo para que ocupe toda la pantalla del usuario.**
- `setLocationRelativeTo(parent)` **posiciona el diálogo en el centro de la ventana principal (pasada como** `parent`**).**

**Panel principal:**

- `JPanel mainPanel = new JPanel(new GridLayout(6, 1, 10, 10));` **crea un panel con un** `GridLayout` **que organiza sus componentes en una cuadrícula de 6 filas y 1 columna. Los valores** `10, 10` **representan el espacio horizontal y vertical entre los componentes.**

**Título centrado:**

- `JLabel tituloLabel = new JLabel("El Mirador", SwingConstants.CENTER);` **crea una etiqueta (**`JLabel`**) con el texto "El Mirador", centrado horizontalmente.**
- `tituloLabel.setFont(new Font("Serif", Font.BOLD, 24));` **establece el tipo de fuente como "Serif", en negrita y tamaño 24.**
- `tituloLabel.setForeground(Color.WHITE);` **cambia el color del texto a blanco.**

**Fondo para el título:**

- `tituloLabel.setOpaque(true);` **hace que el fondo de la etiqueta sea visible.**
- `tituloLabel.setBackground(new Color(34, 45, 50));` **establece el fondo de la etiqueta con un color oscuro (**`Color(34, 45, 50)`**).**

**Sombra en el texto:**

- `tituloLabel.setText("<html><span style='text-shadow: 2px 2px 8px #000000;'>El Mirador</span></html>");` **agrega una sombra al texto de la etiqueta, usando HTML para aplicar el estilo de la sombra.**

**Borde decorativo:**

- `tituloLabel.setBorder(BorderFactory.createLineBorder(new Color(255, 150, 0), 2));` **crea un borde de 2 píxeles alrededor de la etiqueta con un color naranja claro (**`Color(255, 150, 0)`**).**

**Agregando el título al panel:**

- **`mainPanel.add(tituloLabel, BorderLayout.NORTH);` agrega la etiqueta `tituloLabel` al panel `mainPanel`, alineado en la parte superior del panel (`BorderLayout.NORTH`).**

```
public AltaReservasDialog(JFrame parent) {
    super(parent, "Alta Reservas", true);
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setSize(screenSize.width, screenSize.height);
    setLocationRelativeTo(parent);
    // Panel principal con GridLayout
    JPanel mainPanel = new JPanel(new GridLayout(6, 1, 10, 10));

 // Título centrado
    JLabel tituloLabel = new JLabel("El Mirador", SwingConstants.CENTER);
    tituloLabel.setFont(new Font("Serif", Font.BOLD, 24)); // Mantener el tamaño de fuente original
    tituloLabel.setForeground(Color.WHITE); // Color blanco para el texto

    // Fondo para el título
    tituloLabel.setOpaque(true); // Para que el fondo se vea
    tituloLabel.setBackground(new Color(34, 45, 50)); // Fondo oscuro para el título

    // Agregar una sombra al texto (sin cambiar la estructura del panel)
    tituloLabel.setText("<html><span style='text-shadow: 2px 2px 8px #000000;'>El Mirador</span></html>");

    // Borde decorativo alrededor del título
    tituloLabel.setBorder(BorderFactory.createLineBorder(new Color(255, 150, 0), 2)); // Borde naranja claro

    // Agregar el título al panel
    mainPanel.add(tituloLabel, BorderLayout.NORTH);
```

---

## 1. Creación del Panel de Datos Personales:

- `JPanel personalDataPanel = new JPanel();` crea un nuevo panel.
- `personalDataPanel.setBorder(new TitledBorder("Datos Personales"));` agrega un borde al panel con el título "Datos Personales".
- `personalDataPanel.setBackground(Color.CYAN);` establece el color de fondo del panel como cian.
- `personalDataPanel.setLayout(new GridLayout(4, 2, 5, 5));` define que el panel usará un `GridLayout` con 4 filas y 2 columnas, con un espacio de 5 píxeles entre cada componente.

## 2. Campos de Texto:

- Se crean varios campos de texto (`JTextField`) para que el usuario ingrese su nombre, apellidos, DNI, teléfono e importe.

- `nombreField = new JTextField();` crea un campo de texto para el nombre.
- `apellidosField = new JTextField();` crea un campo de texto para los apellidos.
- `dniField = new JTextField();` crea un campo de texto para el DNI.
- `telefonoField = new JTextField();` crea un campo de texto para el teléfono.
- `importeField = new JTextField();` crea un campo de texto para el importe, pero este es solo de lectura, por lo que:
  - `importeField.setEditable(false);` hace que no se pueda editar.
  - `importeField.setBackground(Color.MAGENTA);` cambia el fondo a magenta.
  - `importeField.setForeground(Color.BLACK);` cambia el color del texto a negro.

### 3. Método `setTextFieldBackground`:

- `setTextFieldBackground(nombreField, Color.CYAN);` cambia el fondo de los campos de texto (nombre, apellidos, DNI y teléfono) al mismo color cian que el del panel.

### 4. Validación de Entrada para DNI y Teléfono:

- **DNI**:
  - Se agrega un `KeyListener` al campo `dniField` para interceptar las pulsaciones de teclas.
  - `dniField.getText().length() >= 9` valida que el texto no exceda los 9 caracteres.
  - `!Character.isLetterOrDigit(e.getKeyChar())` valida que solo se ingresen caracteres alfanuméricos.
  - Si no se cumple alguna de estas condiciones, se cancela la entrada (`e.consume()`) y se muestra un mensaje de error con `JOptionPane.showMessageDialog()`, indicando que el DNI debe tener 8 números y 1 letra.
- **Teléfono**:
  - Se agrega un `KeyListener` al campo `telefonoField` con validaciones similares para asegurar que el número de teléfono tenga exactamente 9 dígitos.
  - `telefonoField.getText().length() >= 9` valida que el texto no exceda los 9 dígitos.
  - `!Character.isDigit(e.getKeyChar())` valida que solo se ingresen dígitos.

- Si no se cumple alguna de estas condiciones, se cancela la entrada y se muestra un mensaje de error indicando que el teléfono debe tener 9 dígitos.

## 5. Agregando los Componentes al Panel:

- `personalDataPanel.add(new JLabel("Nombre:"));` agrega una etiqueta con el texto "Nombre:" al panel.
- `personalDataPanel.add(nombreField);` agrega el campo de texto para el nombre al panel.
- Repite este proceso para los campos de apellidos, DNI y teléfono.
- Finalmente, todos los componentes se colocan en el panel siguiendo el layout definido por el `GridLayout`, lo que garantiza que se ubiquen en una cuadrícula de 4 filas y 2 columnas.

```
// Panel de Datos Personales
JPanel personalDataPanel = new JPanel();
personalDataPanel.setBorder(new TitledBorder("Datos Personales"));
personalDataPanel.setBackground(Color.CYAN);
personalDataPanel.setLayout(new GridLayout(4, 2, 5, 5));

nombreField = new JTextField();
apellidosField = new JTextField();
dniField = new JTextField();
telefonoField = new JTextField();
importeField = new JTextField();
importeField.setEditable(false);
importeField.setBackground(Color.MAGENTA);
importeField.setForeground(Color.BLACK);

setTextFieldBackground(nombreField, Color.CYAN);
setTextFieldBackground(apellidosField, Color.CYAN);
setTextFieldBackground(dniField, Color.CYAN);
setTextFieldBackground(telefonoField, Color.CYAN);

dniField.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        if (dniField.getText().length() >= 9 || !Character.isLetterOrDigit(e.getKeyChar())) {
            e.consume();
            JOptionPane.showMessageDialog(null, "El DNI debe tener 8 números y 1 letra.");
        }
    }
});

telefonoField.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        if (telefonoField.getText().length() >= 9 || !Character.isDigit(e.getKeyChar())) {
            e.consume();
            JOptionPane.showMessageDialog(null, "El Teléfono debe tener 9 dígitos.");
        }
    }
});

personalDataPanel.add(new JLabel("Nombre:"));
personalDataPanel.add(nombreField);
personalDataPanel.add(new JLabel("Apellidos:"));
personalDataPanel.add(apellidosField);
personalDataPanel.add(new JLabel("DNI:"));
personalDataPanel.add(dniField);
personalDataPanel.add(new JLabel("Teléfono:"));
personalDataPanel.add(telefonoField);
```

---

### Creación del Panel de Fechas:

- `JPanel datePanel = new JPanel();` crea un nuevo panel.
- `datePanel.setBorder(new TitledBorder("Fechas"));` agrega un borde al panel con el título "Fechas".
- `datePanel.setBackground(Color.GREEN);` establece el color de fondo del panel como verde.
- `datePanel.setLayout(new GridLayout(3, 2, 5, 5));` define un `GridLayout` de 3 filas y 2 columnas, con un espacio de 5 píxeles entre los componentes.

### 2. Creación de Componentes de Fechas:

- **Spinners para las Fechas**:
  - `fechaEntradaSpinner = new JSpinner(new SpinnerDateModel());` crea un `JSpinner` para seleccionar la fecha de entrada. Se utiliza un `SpinnerDateModel` que permite seleccionar fechas.
  - `fechaSalidaSpinner = new JSpinner(new SpinnerDateModel());` crea un `JSpinner` similar para seleccionar la fecha de salida.
- **Campo de Texto para los Días de Estancia**:
  - `diasEstanciaField = new JTextField();` crea un campo de texto para mostrar el número de días de estancia.
  - `diasEstanciaField.setEditable(false);` hace que el campo de texto no sea editable, ya que su valor será calculado automáticamente.

### 3. Configuración de Fondos:

- `setSpinnerBackground(fechaEntradaSpinner, Color.GREEN);` cambia el fondo del spinner de fecha de entrada a verde.
- `setSpinnerBackground(fechaSalidaSpinner, Color.GREEN);` cambia el fondo del spinner de fecha de salida a verde.
- `setTextFieldBackground(diasEstanciaField, Color.GREEN);` cambia el fondo del campo de texto de los días de estancia a verde.

### 4. Eventos de Cambio en los Spinners:

- `fechaEntradaSpinner.addChangeListener(e -> calcularDiasEstancia());` y `fechaSalidaSpinner.addChangeListener(e -> calcularDiasEstancia());` agregan un `ChangeListener` a ambos spinners.

Cada vez que el valor de cualquiera de los spinners cambia (es decir, se selecciona una nueva fecha), se ejecuta el método `calcularDiasEstancia()`. Este método se utilizará para calcular la cantidad de días entre las fechas seleccionadas y mostrar el resultado en el campo de texto `diasEstanciaField`.

## 5. Agregando Componentes al Panel:

- `datePanel.add(new JLabel("Fecha Entrada:"));` agrega una etiqueta con el texto "Fecha Entrada:" al panel.
- `datePanel.add(fechaEntradaSpinner);` agrega el spinner para la fecha de entrada al panel.
- `datePanel.add(new JLabel("Fecha Salida:"));` agrega una etiqueta con el texto "Fecha Salida:" al panel.
- `datePanel.add(fechaSalidaSpinner);` agrega el spinner para la fecha de salida al panel.
- `datePanel.add(new JLabel("Nº de días estancia:"));` agrega una etiqueta con el texto "Nº de días estancia:" al panel.
- `datePanel.add(diasEstanciaField);` agrega el campo de texto donde se mostrará el número de días de estancia al panel.

```
// Panel de Fechas
JPanel datePanel = new JPanel();
datePanel.setBorder(new TitledBorder("Fechas"));
datePanel.setBackground(Color.GREEN);
datePanel.setLayout(new GridLayout(3, 2, 5, 5));

fechaEntradaSpinner = new JSpinner(new SpinnerDateModel());
fechaSalidaSpinner = new JSpinner(new SpinnerDateModel());
diasEstanciaField = new JTextField();
diasEstanciaField.setEditable(false);

setSpinnerBackground(fechaEntradaSpinner, Color.GREEN);
setSpinnerBackground(fechaSalidaSpinner, Color.GREEN);
setTextFieldBackground(diasEstanciaField, Color.GREEN);

fechaEntradaSpinner.addChangeListener(e -> calcularDiasEstancia());
fechaSalidaSpinner.addChangeListener(e -> calcularDiasEstancia());

datePanel.add(new JLabel("Fecha Entrada:"));
datePanel.add(fechaEntradaSpinner);
datePanel.add(new JLabel("Fecha Salida:"));
datePanel.add(fechaSalidaSpinner);
datePanel.add(new JLabel("Nº de días estancia:"));
datePanel.add(diasEstanciaField);
```

---

## Panel de Habitación (`roomPanel`):

- `JPanel roomPanel = new JPanel();` crea un nuevo panel para la habitación.
- `roomPanel.setBorder(new TitledBorder("Habitación"));` establece un borde con el título "Habitación".
- `roomPanel.setBackground(Color.MAGENTA);` define el color de fondo como magenta.
- `roomPanel.setLayout(new GridLayout(6, 2, 5, 5));` organiza los componentes en un `GridLayout` de 6 filas y 2 columnas con 5 píxeles de separación entre los componentes.

## 2. Componentes del Panel de Habitación:

- **ComboBox para Tipo de Habitación**:
  - `tipoHabitacionCombo = new JComboBox<>(new String[]{"", "Simple", "Doble", "Suite"});` crea un combo box con las opciones de tipo de habitación (Simple, Doble, Suite).
  - `tipoHabitacionCombo.setBackground(Color.MAGENTA);` cambia el fondo del combo box a magenta.
- **Spinner para Número de Habitaciones**:
  - `numHabitacionesSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 50, 1));` crea un `JSpinner` para seleccionar el número de habitaciones, con valores entre 0 y 50.
- **Checkbox para Niños**:
  - `ninosCheckBox = new JCheckBox("¿Niños?");` crea un checkbox para indicar si hay niños.
  - `ninosCheckBox.setBackground(Color.MAGENTA);` establece el fondo del checkbox a magenta.
- **Spinner para Edad de los Niños**:
  - `edadNinosSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 14, 1));` crea un `JSpinner` para seleccionar la edad de los niños, con un rango de 0 a 14 años.
- **Campo de Texto para Extras**:
  - `extrasField = new JTextField();` crea un campo de texto donde se pueden ingresar los extras asociados a la habitación.
- **Configuración de Fondo**:
  - `setSpinnerBackground(numHabitacionesSpinner, Color.MAGENTA);` y `setSpinnerBackground(edadNinosSpinner, Color.MAGENTA);` cambian el fondo de los spinners a magenta.

    - `setTextFieldBackground(extrasField, Color.MAGENTA);` cambia el fondo del campo de texto a magenta.

### 3. Habilitar/Deshabilitar Elementos con el Checkbox:

- `ninosCheckBox.addActionListener(e -> { ... });` agrega un `ActionListener` al checkbox para habilitar o deshabilitar los campos relacionados con los niños (`edadNinosSpinner` y `extrasField`) cuando se selecciona o deselecciona el checkbox.
- Si el checkbox está seleccionado (`habilitar`), se habilitan los campos de edad de los niños y los extras, y si no está seleccionado, se deshabilitan.

### 4. Métodos de Cálculo:

- `edadNinosSpinner.addChangeListener(e -> actualizarExtras());` agrega un `ChangeListener` al spinner de edad de los niños. Cuando cambia el valor, se ejecuta el método `actualizarExtras()`.
- `tipoHabitacionCombo.addActionListener(e -> calcularImporte());` agrega un `ActionListener` al combo box de tipo de habitación. Cuando cambia el tipo de habitación seleccionado, se ejecuta el método `calcularImporte()`.
- `numHabitacionesSpinner.addChangeListener(e -> calcularImporte());` agrega un `ChangeListener` al spinner de número de habitaciones para actualizar el importe cada vez que cambia el número de habitaciones.

### 5. Agregando Componentes al Panel:

Los elementos que se agregan al panel de habitación incluyen:

- **Etiquetas y Campos de Entrada**:
    - `roomPanel.add(new JLabel("Tipo de habitación:"));` agrega una etiqueta con el texto "Tipo de habitación:".
    - `roomPanel.add(tipoHabitacionCombo);` agrega el combo box para seleccionar el tipo de habitación.
    - `roomPanel.add(new JLabel("Nº de habitaciones:"));` agrega una etiqueta para el número de habitaciones.
    - `roomPanel.add(numHabitacionesSpinner);` agrega el spinner para el número de habitaciones.
    - `roomPanel.add(new JLabel("Edad de niños (si aplica):"));` agrega una etiqueta para la edad de los niños.
    - `roomPanel.add(edadNinosSpinner);` agrega el spinner para la edad de los niños.

    - `roomPanel.add(new JLabel("Extras:"));` agrega una etiqueta para los extras.
    - `roomPanel.add(extrasField);` agrega el campo de texto para los extras.
    - `roomPanel.add(new JLabel("Importe habitación (€):"));` agrega una etiqueta para el importe de la habitación.
    - `roomPanel.add(importeField);` agrega el campo de texto para mostrar el importe de la habitación.
    - `roomPanel.add(ninosCheckBox);` agrega el checkbox para seleccionar si hay niños.

### 6. Panel Principal:

- Se agrega el panel de habitación al `mainPanel` junto con otros paneles previamente configurados:
    - `mainPanel.add(personalDataPanel);`
    - `mainPanel.add(datePanel);`
    - `mainPanel.add(roomPanel);`
- Se añade el `mainPanel` al contenedor principal con `add(mainPanel, BorderLayout.CENTER);`.

### 7. Botón de Aceptar:

- Se crea un panel de botones con `JPanel buttonPanel = new JPanel();`.
- Se agrega un botón de "Aceptar" con `JButton btnAceptar = new JButton("Aceptar");`.
- Se agrega un `ActionListener` al botón que cierra el diálogo cuando se hace clic con `btnAceptar.addActionListener(e -> dispose());`.
- Finalmente, se agrega el botón al panel y el panel de botones al contenedor principal con `add(buttonPanel, BorderLayout.SOUTH);`.

### 8. Uso de JTabbedPane:

- Se crea un `JTabbedPane` con `JTabbedPane tabbedPane = new JTabbedPane();`.
- Dentro de este `JTabbedPane`, se agregan dos paneles: `clientePanel` y `habitacionPanel`, que contienen etiquetas para mostrar los datos ingresados en los otros paneles.

```java
// Panel de Habitación
JPanel roomPanel = new JPanel();
roomPanel.setBorder(new TitledBorder("Habitación"));
roomPanel.setBackground(Color.MAGENTA);
roomPanel.setLayout(new GridLayout(6, 2, 5, 5));

tipoHabitacionCombo = new JComboBox<>(new String[]{"","Simple", "Doble", "Suite"});
numHabitacionesSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 50, 1));
tipoHabitacionCombo.setBackground(Color.MAGENTA);
ninosCheckBox = new JCheckBox("¿Niños?");
ninosCheckBox.setBackground(Color.MAGENTA);

edadNinosSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 14, 1));
extrasField = new JTextField();

setSpinnerBackground(numHabitacionesSpinner, Color.MAGENTA);
setSpinnerBackground(edadNinosSpinner, Color.MAGENTA);
setTextFieldBackground(extrasField, Color.MAGENTA);

edadNinosSpinner.setEnabled(false);
extrasField.setEnabled(false);

ninosCheckBox.addActionListener(e -> {
    boolean habilitar = ninosCheckBox.isSelected();
    edadNinosSpinner.setEnabled(habilitar);
    extrasField.setEnabled(habilitar);
});

edadNinosSpinner.addChangeListener(e -> actualizarExtras());

tipoHabitacionCombo.addActionListener(e -> calcularImporte());
numHabitacionesSpinner.addChangeListener(e -> calcularImporte());

roomPanel.add(new JLabel("Tipo de habitación:"));
roomPanel.add(tipoHabitacionCombo);
roomPanel.add(new JLabel("Nº de habitaciones:"));
roomPanel.add(numHabitacionesSpinner);
roomPanel.add(new JLabel("Edad de niños (si aplica):"));
roomPanel.add(edadNinosSpinner);
roomPanel.add(new JLabel("Extras:"));
roomPanel.add(extrasField);
roomPanel.add(new JLabel("Importe habitación (€):"));
roomPanel.add(importeField);
roomPanel.add(ninosCheckBox);

mainPanel.add(personalDataPanel);
mainPanel.add(datePanel);
mainPanel.add(roomPanel);
add(mainPanel, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();
JButton btnAceptar = new JButton("Aceptar");
btnAceptar.addActionListener(e -> dispose());
buttonPanel.add(btnAceptar);
add(buttonPanel, BorderLayout.SOUTH);

JTabbedPane tabbedPane = new JTabbedPane();
```

```java
JTabbedPane tabbedPane = new JTabbedPane();

JPanel clientePanel = new JPanel();
clientePanel.setLayout(new GridLayout(4, 2, 5, 5));
clientePanel.setBorder(new TitledBorder("Datos del Cliente"));
JLabel lblNombre = new JLabel("Nombre:");
JLabel lblNombreValue = new JLabel();
JLabel lblApellidos = new JLabel("Apellidos:");
JLabel lblApellidosValue = new JLabel();
JLabel lblDni = new JLabel("DNI:");
JLabel lblDniValue = new JLabel();
JLabel lblTelefono = new JLabel("Teléfono:");
JLabel lblTelefonoValue = new JLabel();

clientePanel.add(lblNombre);
clientePanel.add(lblNombreValue);
clientePanel.add(lblApellidos);
clientePanel.add(lblApellidosValue);
clientePanel.add(lblDni);
clientePanel.add(lblDniValue);
clientePanel.add(lblTelefono);
clientePanel.add(lblTelefonoValue);

JPanel habitacionPanel = new JPanel();
habitacionPanel.setLayout(new GridLayout(4, 2, 5, 5));
habitacionPanel.setBorder(new TitledBorder("Datos de la Habitación"));

JLabel lblTipoHabitacion = new JLabel("Tipo de Habitación:");
JLabel lblTipoHabitacionValue = new JLabel();
JLabel lblNumHabitaciones = new JLabel("Nº de Habitaciones:");
JLabel lblNumHabitacionesValue = new JLabel();
JLabel lblExtras = new JLabel("Extras:");
JLabel lblExtrasValue = new JLabel();
JLabel lblImporte = new JLabel("Importe Total (€):");
JLabel lblImporteValue = new JLabel();

habitacionPanel.add(lblTipoHabitacion);
habitacionPanel.add(lblTipoHabitacionValue);
habitacionPanel.add(lblNumHabitaciones);
habitacionPanel.add(lblNumHabitacionesValue);
habitacionPanel.add(lblExtras);
habitacionPanel.add(lblExtrasValue);
habitacionPanel.add(lblImporte);
habitacionPanel.add(lblImporteValue);
```

---

## Creación del Botón y Establecimiento de Propiedades:

- `JButton btnImprimir = new JButton("Imprimir");` crea el botón con el texto "Imprimir".
- `ImageIcon imprimirIcon = new ImageIcon(getClass().getResource("/Media/imprimir.png"));` carga un ícono desde el archivo ubicado en la carpeta `/Media`. Este archivo debe estar en el directorio adecuado del proyecto, generalmente dentro de `src` o en la estructura de recursos del proyecto.

- `btnImprimir.setIcon(new ImageIcon(imprimirIcon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH)));` ajusta el tamaño del ícono a 20x20 píxeles, asegurándose de que se vea suavizado cuando se redimensiona.
- `btnImprimir.setPreferredSize(new Dimension(110, 30));` establece un tamaño preferido de 110 píxeles de ancho y 30 píxeles de alto para el botón, asegurando que el botón tenga un tamaño adecuado para contener el texto y el ícono.

## 2. Acción al Hacer Clic en el Botón:

El botón tiene un `ActionListener` que se activa cuando el usuario hace clic en él. El `ActionListener` realiza las siguientes acciones:

- **Validar los Campos**:
  - `if (validarCampos())` verifica si todos los campos necesarios están completos y son válidos. El método `validarCampos()` debe ser definido en otra parte del código, y se espera que devuelva `true` si todos los campos obligatorios están correctamente llenados, y `false` si falta algún dato.
- **Actualizar las Etiquetas de Datos**: Si la validación es exitosa, los valores de los campos de entrada se muestran en las etiquetas correspondientes del cuarto panel (probablemente un panel de resumen o confirmación). Esto se hace utilizando `lblNombreValue.setText(nombreField.getText());`, y lo mismo ocurre para los demás campos:
  - `lblNombreValue.setText(nombreField.getText());`
  - `lblApellidosValue.setText(apellidosField.getText());`
  - `lblDniValue.setText(dniField.getText());`
  - `lblTelefonoValue.setText(telefonoField.getText());`
  - `lblTipoHabitacionValue.setText((String) tipoHabitacionCombo.getSelectedItem());`
  - `lblNumHabitacionesValue.setText(String.valueOf(numHabitacionesSpinner.getValue()));`
  - `lblExtrasValue.setText(extrasField.getText());`
  - `lblImporteValue.setText(importeField.getText());`
- Cada llamada a `setText` actualiza el texto mostrado en las etiquetas para reflejar los valores que el usuario ha ingresado.
- **Mostrar un Mensaje de Confirmación**:
  - `JOptionPane.showMessageDialog(this, "Datos impresos correctamente en las pestañas.");` muestra un cuadro de mensaje indicando que los datos se han "impreso" correctamente en las pestañas, lo que podría referirse a la actualización de la interfaz con la información proporcionada por el usuario.

- **Manejo de Error**: Si los campos no están completos o no son válidos, se muestra un mensaje de error utilizando:
    - `JOptionPane.showMessageDialog(this, "Por favor, complete todos los campos obligatorios.", "Error", JOptionPane.ERROR_MESSAGE);` este mensaje informa al usuario que debe completar todos los campos necesarios para continuar.

```
    JButton btnImprimir = new JButton("Imprimir");
// Cargar el ícono desde la carpeta Media
ImageIcon imprimirIcon = new ImageIcon(getClass().getResource("/Media/imprimir.png"));
btnImprimir.setIcon(new ImageIcon(imprimirIcon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH))); // Ajusta el tamaño del ícono
btnImprimir.setPreferredSize(new Dimension(110, 30)); // Establece un tamaño preferido para el botón


    btnImprimir.addActionListener(e -> {
        if (validarCampos()) {
            // Actualizar las etiquetas del cuarto panel
            lblNombreValue.setText(nombreField.getText());
            lblApellidosValue.setText(apellidosField.getText());
            lblDniValue.setText(dniField.getText());
            lblTelefonoValue.setText(telefonoField.getText());
            lblTipoHabitacionValue.setText((String) tipoHabitacionCombo.getSelectedItem());
            lblNumHabitacionesValue.setText(String.valueOf(numHabitacionesSpinner.getValue()));
            lblExtrasValue.setText(extrasField.getText());
            lblImporteValue.setText(importeField.getText());

            JOptionPane.showMessageDialog(this, "Datos impresos correctamente en las pestañas.");
        } else {
            JOptionPane.showMessageDialog(this, "Por favor, complete todos los campos obligatorios.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    });
```

---

## Creación del Botón y Establecimiento de Propiedades:

- `JButton btnNuevo = new JButton("Nuevo");` crea el botón con el texto "Nuevo".
- `ImageIcon nuevoIcon = new ImageIcon(getClass().getResource("/Media/nuevo.png"));` carga un ícono desde el archivo ubicado en la carpeta `/Media`. Este archivo debe estar en el directorio adecuado del proyecto, generalmente dentro de `src` o en la estructura de recursos del proyecto.
- `btnNuevo.setIcon(new ImageIcon(nuevoIcon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH)));` ajusta el tamaño del ícono a 20x20 píxeles, asegurándose de que se vea suavizado cuando se redimensiona.
- `btnNuevo.setPreferredSize(new Dimension(110, 30));` establece un tamaño preferido de 110 píxeles de ancho y 30 píxeles de alto para el botón, asegurando que el botón tenga un tamaño adecuado para contener el texto y el ícono.

## 2. Acción al Hacer Clic en el Botón:

El botón tiene un `ActionListener` que se activa cuando el usuario hace clic en él. Este `ActionListener` realiza las siguientes acciones:

- **Reiniciar los Campos del Formulario**: El evento de clic reinicia todos los campos a sus valores por defecto o vacíos:
    - `nombreField.setText("");` reinicia el campo de nombre.
    - `apellidosField.setText("");` reinicia el campo de apellidos.
    - `dniField.setText("");` reinicia el campo de DNI.
    - `telefonoField.setText("");` reinicia el campo de teléfono.
    - `diasEstanciaField.setText("");` reinicia el campo de días de estancia.
    - `extrasField.setText("");` reinicia el campo de extras.
    - `importeField.setText("0");` establece el valor del importe a "0".
    - `tipoHabitacionCombo.setSelectedIndex(0);` restablece el combo box de tipo de habitación a la primera opción (vacía).
    - `numHabitacionesSpinner.setValue(0);` reinicia el número de habitaciones a 0.
    - `edadNinosSpinner.setValue(0);` restablece el spinner de edad de niños a 0.
    - `ninosCheckBox.setSelected(false);` desmarca la casilla de niños.
    - `fechaEntradaSpinner.setValue(new Date());` restablece la fecha de entrada al valor actual.
    - `fechaSalidaSpinner.setValue(new Date());` restablece la fecha de salida al valor actual.
- **Deshabilitar los Campos de Niños y Extras**:
    - `edadNinosSpinner.setEnabled(false);` deshabilita el spinner de edad de niños, ya que se supone que no es necesario si no se seleccionan niños.
    - `extrasField.setEnabled(false);` deshabilita el campo de extras, ya que no es necesario sin niños.
- **Restablecer el Foco en el Campo de Nombre**:
    - `nombreField.requestFocus();` coloca el foco en el campo de nombre, de modo que el usuario puede empezar a escribir inmediatamente al hacer clic en "Nuevo".

```java
// Crear el botón btnNuevo con el ícono y el texto
    JButton btnNuevo = new JButton("Nuevo");
    // Cargar el ícono desde la carpeta Media
    ImageIcon nuevoIcon = new ImageIcon(getClass().getResource("/Media/nuevo.png"));
    btnNuevo.setIcon(new ImageIcon(nuevoIcon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH))); // Ajusta el tamaño del ícono
    btnNuevo.setPreferredSize(new Dimension(110, 30)); // Establece un tamaño preferido para el botón

    btnNuevo.addActionListener(e -> {
        // Reiniciar todos los campos del formulario
        nombreField.setText("");
        apellidosField.setText("");
        dniField.setText("");
        telefonoField.setText("");
        diasEstanciaField.setText("");
        extrasField.setText("");
        importeField.setText("0");
        tipoHabitacionCombo.setSelectedIndex(0);
        numHabitacionesSpinner.setValue(0);
        edadNinosSpinner.setValue(0);
        ninosCheckBox.setSelected(false);
        fechaEntradaSpinner.setValue(new Date());
        fechaSalidaSpinner.setValue(new Date());
        edadNinosSpinner.setEnabled(false);
        extrasField.setEnabled(false);
        nombreField.requestFocus();
    });
```

---

## Creación del Botón y Establecimiento de Propiedades:

- `JButton btnGuardar = new JButton("Guardar");` crea un botón con el texto "Guardar".
- `ImageIcon guardarIcon = new ImageIcon(getClass().getResource("/Media/guardar.png"));` carga un ícono desde el archivo ubicado en la carpeta `/Media`. Asegúrate de que el archivo de imagen esté correctamente ubicado en la carpeta de recursos de tu proyecto.
- `btnGuardar.setIcon(new ImageIcon(guardarIcon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH)));` ajusta el tamaño del ícono a 20x20 píxeles, asegurando que se vea de forma adecuada en el botón.
- `btnGuardar.setPreferredSize(new Dimension(110, 30));` establece un tamaño preferido de 110 píxeles de ancho y 30 píxeles de alto para el botón.

## 2. Acción al Hacer Clic en el Botón:

El botón tiene un `ActionListener` que se activa cuando el usuario hace clic en él. Este `ActionListener` realiza las siguientes acciones:

- **Validación de Campos**:
    - `if (validarCampos()) {` verifica si todos los campos obligatorios son válidos. El método `validarCampos()` debe ser definido previamente para comprobar si los datos del formulario cumplen con los requisitos esperados (por ejemplo, si están completos y son válidos).
- **Mostrar Mensaje de Guardado**:

    - `JOptionPane.showMessageDialog(this, "Registro Guardado.");` muestra un mensaje emergente informando al usuario que los datos han sido guardados correctamente.
- **Mensaje de Error**:
    - Si los campos no son válidos (si `validarCampos()` devuelve `false`), se muestra un mensaje de error:
        - `JOptionPane.showMessageDialog(this, "Por favor, complete todos los campos obligatorios.", "Error", JOptionPane.ERROR_MESSAGE);` muestra una ventana de mensaje con un icono de error y un texto indicando que faltan campos por completar.

```
    // Botón Guardar
    JButton btnGuardar = new JButton("Guardar");
// Cargar el ícono desde la carpeta Media
ImageIcon guardarIcon = new ImageIcon(getClass().getResource("/Media/guardar.png"));
btnGuardar.setIcon(new ImageIcon(guardarIcon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH))); // Ajusta el tamaño del ícono
btnGuardar.setPreferredSize(new Dimension(110, 30)); // Establece un tamaño preferido para el botón


    btnGuardar.addActionListener(e -> {
        if (validarCampos()) {
            JOptionPane.showMessageDialog(this, "Registro Guardado.");
        } else {
            JOptionPane.showMessageDialog(this, "Por favor, complete todos los campos obligatorios.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    });
```

---

## Creación del Panel de Descuento:

- `JPanel descuentoPanel = new JPanel();` crea un nuevo panel para contener los componentes relacionados con el descuento.
- `descuentoPanel.setBorder(new TitledBorder("Descuento Promocional"));` establece un borde con título ("Descuento Promocional") para el panel, lo que proporciona una mejor organización visual.
- `descuentoPanel.setLayout(new GridLayout(2, 1, 5, 5));` establece el layout del panel a un **GridLayout** de 2 filas y 1 columna, con márgenes de 5 píxeles en ambas direcciones.

## 2. Creación y Configuración del JSlider:

- `JSlider descuentoSlider = new JSlider(0, 50, 0);` crea un **JSlider** para ajustar el descuento. Este slider tiene un rango de valores de 0 a 50 (para representar un descuento de entre 0% y 50%).

- `descuentoSlider.setMajorTickSpacing(10);` establece que las marcas principales del slider se ubiquen cada 10 unidades (representando cada 10% de descuento).
- `descuentoSlider.setMinorTickSpacing(5);` establece que las marcas menores se ubiquen cada 5 unidades (para mostrar marcas intermedias cada 5% de descuento).
- `descuentoSlider.setPaintTicks(true);` activa la visualización de las marcas en el slider.
- `descuentoSlider.setPaintLabels(true);` activa la visualización de las etiquetas numéricas en el slider (0%, 5%, 10%, etc.).
- `descuentoSlider.setToolTipText("Ajusta el porcentaje de descuento aplicable a la reserva.");` establece un **tooltip** que aparece al pasar el ratón por encima del slider, explicando su función.

### 3. Creación del JLabel para Mostrar el Descuento Aplicado:

- `JLabel descuentoLabel = new JLabel("Descuento aplicado: 0%");` crea un **JLabel** que mostrará el valor del descuento aplicado. Inicialmente se muestra "0%" ya que el slider está en 0%.
- `descuentoLabel.setHorizontalAlignment(SwingConstants.CENTER);` centra el texto dentro del JLabel para una mejor visualización.

### 4. Evento para Actualizar el Descuento en Tiempo Real:

- `descuentoSlider.addChangeListener(e -> { ... });` agrega un **ChangeListener** al slider para escuchar los cambios en su valor.
- Dentro del `ChangeListener`, se obtiene el valor actual del slider mediante `descuentoSlider.getValue();`, y luego se actualiza el texto del **JLabel** para mostrar el porcentaje de descuento.
- Se llama al método `calcularImporteConDescuento(descuento);` para recalcular el importe con el nuevo descuento. Este método debe estar implementado en tu código para aplicar el descuento al precio de la reserva.

### 5. Añadir Componentes al Panel y al Layout Principal:

- `descuentoPanel.add(descuentoLabel);` y `descuentoPanel.add(descuentoSlider);` añaden el **JLabel** y el **JSlider** al panel de descuento.
- `mainPanel.add(descuentoPanel);` añade el panel de descuento al **mainPanel**, que es el contenedor principal de tu interfaz.

```java
// Panel de Descuento
JPanel descuentoPanel = new JPanel();
descuentoPanel.setBorder(new TitledBorder("Descuento Promocional"));
descuentoPanel.setLayout(new GridLayout(2, 1, 5, 5));

JSlider descuentoSlider = new JSlider(0, 50, 0); // Descuento del 0% al 50%
descuentoSlider.setMajorTickSpacing(10);
descuentoSlider.setMinorTickSpacing(5);
descuentoSlider.setPaintTicks(true);
descuentoSlider.setPaintLabels(true);
descuentoSlider.setToolTipText("Ajusta el porcentaje de descuento aplicable a la reserva.");

JLabel descuentoLabel = new JLabel("Descuento aplicado: 0%");
descuentoLabel.setHorizontalAlignment(SwingConstants.CENTER);

// Evento para actualizar el importe en tiempo real
descuentoSlider.addChangeListener(e -> {
    int descuento = descuentoSlider.getValue();
    descuentoLabel.setText("Descuento aplicado: " + descuento + "%");
    calcularImporteConDescuento(descuento);
});

// Añadir el slider y el label al panel
descuentoPanel.add(descuentoLabel);
descuentoPanel.add(descuentoSlider);

// Añadir el panel de descuento al layout principal
mainPanel.add(descuentoPanel);
```

---

## Funcionamiento:

1. **Verificación del Importe Inicial**: Primero, el método verifica si el importe original es válido (es decir, mayor que cero). Si no es válido, el campo que muestra el importe se ajusta a cero y el cálculo no se realiza.
2. **Cálculo del Importe con Descuento**: Si el importe es válido, el método calcula el nuevo importe aplicando el descuento. Este descuento se calcula restando un porcentaje del importe original. El porcentaje de descuento es determinado por el valor del control deslizante (el slider), que permite al usuario seleccionar un valor entre 0 y 50%.
3. **Actualización del Importe**: Una vez calculado el importe con descuento, el valor resultante se actualiza en el campo donde se muestra el importe, de manera que el usuario vea el nuevo valor con el descuento aplicado.
4. **Manejo de Errores**: Si ocurre algún error en el proceso de cálculo, como si el importe no es un número válido, el método ajusta el valor a cero para evitar mostrar información incorrecta.

```java
private void calcularImporteConDescuento(int descuento) {
    try {
        if (importeBaseOriginal <= 0) {
            importeField.setText("0");
            return;
        }

        // Aplicar el descuento al importe original
        int importeConDescuento = importeBaseOriginal - (importeBaseOriginal * descuento / 100);

        // Actualizar el campo del importe
        importeField.setText(String.valueOf(importeConDescuento));
    } catch (NumberFormatException e) {
        importeField.setText("0");
    }
}
```

---

El método `calcularImporte` se encarga de calcular el importe total de la reserva de una habitación, teniendo en cuenta varios factores, como el número de días de estancia, el tipo de habitación, el número de habitaciones, y si hay niños que añaden un coste adicional. El cálculo se realiza de la siguiente forma:

## Descripción paso a paso:

1. **Obtener el número de días de estancia**: El número de días de estancia se obtiene del campo de texto `diasEstanciaField`, que es convertido a un valor numérico entero. Si el valor es 0 o negativo, el cálculo se detiene y se muestra un importe de 0.
2. **Obtener el número de habitaciones**: El número de habitaciones se obtiene del `JSpinner` llamado `numHabitacionesSpinner`.
3. **Determinar el precio por día según el tipo de habitación**: Dependiendo de la selección del usuario en el combo box `tipoHabitacionCombo`, se asigna un precio por día:
    - "Simple" → 50€
    - "Doble" → 75€
    - "Suite" → 125€
4. **Calcular el coste base**: El coste base se calcula multiplicando el número de días de estancia, el número de habitaciones y el precio por día de la habitación seleccionada.
5. **Calcular el coste de extras (si aplica)**: Si el usuario selecciona la opción "¿Niños?" mediante el `JCheckBox`, se suman 20€ por cada niño al coste total de la reserva. En este caso, se agrega 20€ por cada día de estancia, independientemente del número de niños.
6. **Sumar el coste base y los extras**: El importe total se obtiene sumando el coste base y los posibles extras.
7. **Guardar el importe base original**: El importe total calculado se guarda en la variable `importeBaseOriginal` para su uso posterior (como el cálculo con descuento).

8. **Actualizar el campo de importe**: Finalmente, el importe total calculado se muestra en el campo de texto `importeField`. Si ocurre algún error en el proceso de conversión o cálculo, el importe se ajusta a cero.

```
private void calcularImporte() {
    try {
        // Obtener el número de días de estancia
        int diasEstancia = Integer.parseInt(diasEstanciaField.getText());
        if (diasEstancia <= 0) {
            importeField.setText("0");
            importeBaseOriginal = 0; // Restablecer el importe original
            return;
        }

        // Obtener el número de habitaciones
        int numHabitaciones = (int) numHabitacionesSpinner.getValue();

        // Determinar el precio por día según el tipo de habitación
        int precioPorDia = switch ((String) tipoHabitacionCombo.getSelectedItem()) {
            case "Simple" -> 50;
            case "Doble" -> 75;
            case "Suite" -> 125;
            default -> 0;
        };

        // Calcular el coste base (habitaciones * días * precio por día)
        int costeBase = diasEstancia * numHabitaciones * precioPorDia;

        // Calcular el coste de extras (20€ por día por habitación si hay niños)
        int costeExtras = 0;
        if (ninosCheckBox.isSelected()) {
            costeExtras = costeExtras + 20;
        }

        // Sumar el coste base y los extras
        int importeTotal = costeBase + costeExtras;

        // Guardar el importe base original
        importeBaseOriginal = importeTotal;

        // Mostrar el importe total en el campo de texto
        importeField.setText(String.valueOf(importeTotal));
    } catch (NumberFormatException e) {
        importeField.setText("0");
        importeBaseOriginal = 0;
    }
}
```

---

## Descripción paso a paso:

1. **Obtener las fechas de entrada y salida**: Las fechas de entrada y salida se obtienen a través de los `JSpinner` llamados `fechaEntradaSpinner` y `fechaSalidaSpinner`, utilizando el modelo de fecha (`SpinnerDateModel`). El método `getDate()` devuelve las fechas seleccionadas como objetos `Date`.
2. **Calcular la diferencia en milisegundos**: Se calcula la diferencia entre las dos fechas en milisegundos utilizando el método `getTime()` de la clase `Date`. La diferencia se almacena en la variable `diferencia`.

3. **Convertir la diferencia a días**: La diferencia en milisegundos se divide por el número de milisegundos que hay en un día (`1000 * 60 * 60 * 24`). Esto convierte la diferencia de tiempo a días completos. El resultado se guarda en la variable `dias`.
4. **Ajustar el número de días**: Si la diferencia es positiva, se suma 1 al número de días, ya que normalmente se considera que la estancia incluye tanto el día de entrada como el día de salida (aunque la fecha de salida no se cuente completamente). Si la diferencia es 0 o negativa (por ejemplo, si la fecha de entrada es posterior a la de salida), se establece un valor mínimo de 1 día de estancia.
5. **Actualizar el campo de texto de días de estancia**: Finalmente, se actualiza el campo de texto `diasEstanciaField` con el número de días calculado.

```
private void calcularDiasEstancia() {
    try {
        Date entrada = ((SpinnerDateModel) fechaEntradaSpinner.getModel()).getDate();
        Date salida = ((SpinnerDateModel) fechaSalidaSpinner.getModel()).getDate();
        long diferencia = salida.getTime() - entrada.getTime();

        // Calcular los días totales
        int dias = (int) (diferencia / (1000 * 60 * 60 * 24));

        // Si la diferencia es positiva, al menos 1 día
        diasEstanciaField.setText(dias > 0 ? String.valueOf(dias + 1) : "1");
    } catch (Exception e) {
        diasEstanciaField.setText("0");
    }
}
```

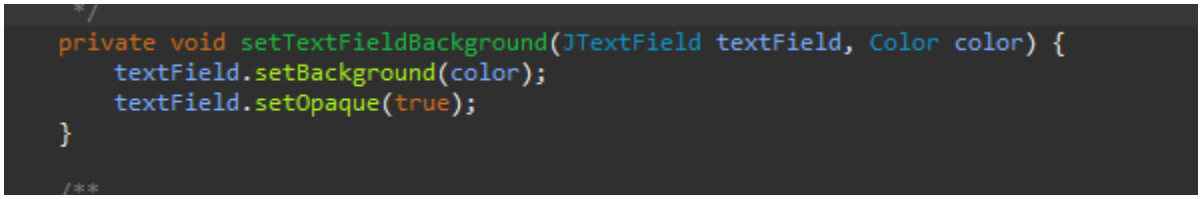
---

**Obtener el editor del `JSpinner`**: Cada `JSpinner` tiene una parte donde se muestra o edita el valor, que se llama "editor". Este editor es como un área de texto o un selector de fechas, y el color de fondo de esta área se puede cambiar.
**Cambiar el color de fondo del editor**: Una vez que se obtiene el editor, se le puede asignar un color de fondo. Esto afecta al área donde el valor es visible y editable.
**Asegurarse de que los botones también cambien de color**: El `JSpinner` tiene botones para aumentar o disminuir el valor. Estos botones están dentro del editor, por lo que, además de cambiar el fondo del área de texto, el método recorre todos los componentes internos del editor (como los botones) y también les cambia el color de fondo. Esto asegura que todos los elementos del `JSpinner` tengan un color uniforme.

```
private void setSpinnerBackground(JSpinner spinner, Color color) {
    JComponent editor = spinner.getEditor();
    editor.setBackground(color);
    for (Component component : editor.getComponents()) {
        component.setBackground(color);
    }
}
```

---

**Cambiar el color de fondo del `JTextField`**: Utiliza el método `setBackground` del `JTextField` para establecer el color de fondo que se desea. Esto afecta al área donde el usuario puede escribir.
**Hacer que el color de fondo sea visible**: El método `setOpaque(true)` se asegura de que el color de fondo sea visible. Por defecto, algunos componentes de la interfaz gráfica tienen la propiedad "opaco" desactivada (es decir, el fondo puede ser transparente). Al ponerlo en `true`, se obliga a que el fondo del `JTextField` sea visible, mostrando el color que se ha asignado.




—----------