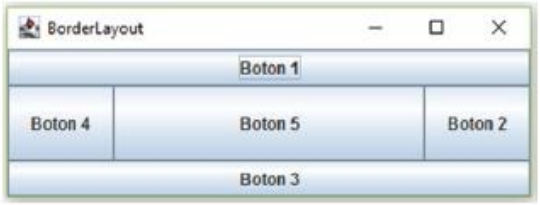
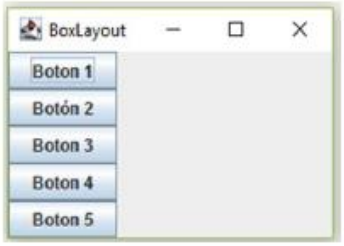
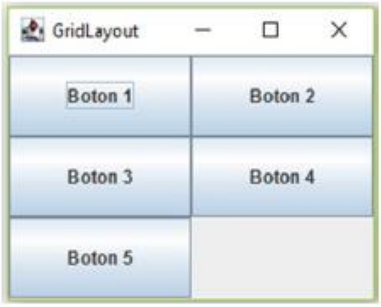
Gestión de contenidos

En una ventana de aplicación deben colocarse numerosos componentes de acuerdo con un cierto orden y disposición gráfica. En Java existen los administradores de diseño (objetos que implementa la interfaz *LayoutManager*) que determinan el tamaño y la posición de los componentes dentro de un contenedor. Aunque estos pueden proporcionar sugerencias de tamaño y alineación, el administrador de diseño de un contenedor tiene la última palabra sobre el tamaño y la posición de los componentes dentro del contenedor. Para su uso se requiere importar el paquete *java.awt.*;*

Tipo	Descripción	Figura
BorderLayout	Establece un contenedor, organizando y redimensionando para que sus componentes se ajusten a cinco regiones: Norte, Sur, Este, Oeste y centro.	
BoxLayout	Permite que varios componentes se distribuyan vertical u horizontalmente.	
GridBagLayout	Alinea los componentes verticalmente, horizontalmente o a lo largo de su línea de base, sin que los componentes sean del mismo tamaño.	
FlowLayout	Organiza los componentes en un flujo direccional, al igual que las líneas de texto en un párrafo.	
CardLayout	Trata cada componente en el contenedor como una tarjeta. Solo se puede ver una tarjeta a la vez y el contenedor actúa como una pila de tarjetas.	
GridLayout	Presenta los componentes de un contenedor en una cuadrícula rectangular. El contenedor se divide en rectángulos de igual tamaño y se coloca un componente en cada rectángulo.	

Objetivos de aprendizaje

Al finalizar este ejercicio, el lector tendrá la capacidad para generar interfaces gráficas de usuario utilizando diferentes *layouts*.

Enunciado: Hotel

Se requiere un programa que permita gestionar el ingreso y salida de los huéspedes de un hotel. El hotel contiene diez habitaciones simples. Las primeras cinco habitaciones tienen un precio de $120 000 por día y las otras cinco habitaciones, $160 000 por día.

El programa cuenta con dos opciones de menú:

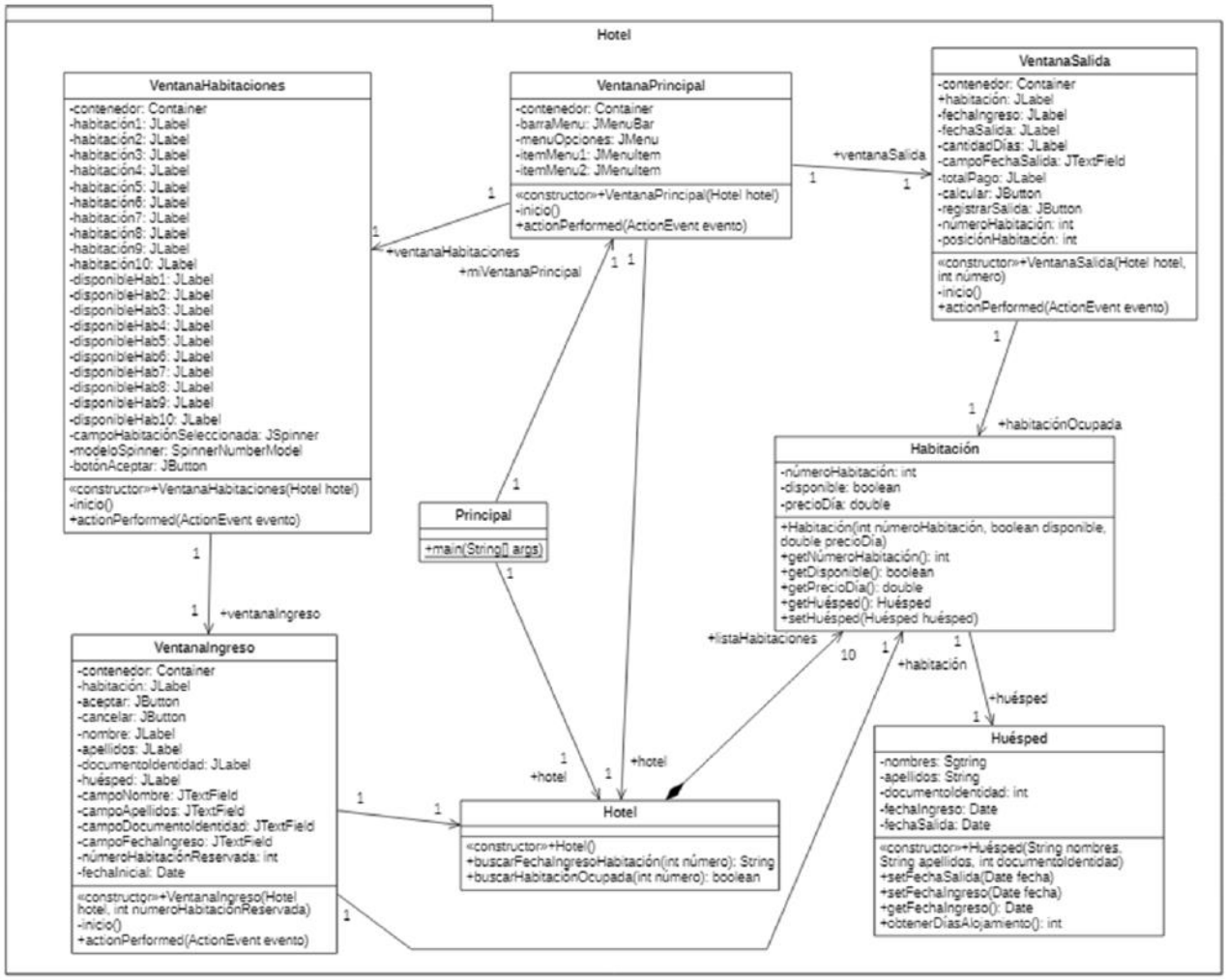
- Consultar habitaciones: al seleccionar este ítem de menú se debe generar una ventana que presenta las diez habitaciones del hotel y su estado: disponible o no disponible. El usuario debe seleccionar el número de la habitación a ocupar. Si se ingresa una habitación ocupada se genera el mensaje de error correspondiente.

  Luego, el programa genera una ventana donde se ingresa la fecha de ingreso y los datos del huésped (nombre, apellidos y número de documento de identidad). Los campos de entrada son obligatorios y se deben validar previamente. Si el registro es correcto se genera el mensaje correspondiente. Si después se consulta el listado de habitaciones, la habitación debe aparecer como “No disponible”.

- Salida de huéspedes: al seleccionar este ítem de menú se debe generar una ventana donde se solicita el número de habitación a entregar. Si la habitación no está ocupada o se ingresa un número o dato incorrecto se genera el mensaje de error correspondiente. Si el número es correcto, se genera una nueva ventana donde se identifica la habitación a entregar y se ingresa la fecha de salida del huésped. Esta fecha debe ser mayor a la fecha de ingreso al hotel. Si la fecha es correcta se habilita un botón que permite calcular el total de días de alojamiento del huésped y el total a pagar por el mismo. Si el usuario oprime el botón “Registrar salida”, la habitación queda disponible.

Se deben utilizar los *layouts* que se consideren convenientes para organizar las diferentes configuraciones de los componentes visuales de las ventanas del programa.

Diagrama de clases



Explicación del diagrama de clases

Se ha definido un paquete denominado “Hotel” que incluye un conjunto de clases. El punto de entrada al programa es la clase Principal que cuenta con el método *main*. La clase Principal está relacionada mediante una relación de asociación con las clases VentanaPrincipal y Hotel.

La clase Hotel tiene un constructor y métodos para buscar la fecha de ingreso de una habitación a partir de su número de habitación y determinar si una habitación está ocupada a partir de su número de habitación. La clase Hotel está compuesta por diez habitaciones, lo cual se presenta por medio de la relación de composición (línea continua entre Hotel y Habitación con un rombo negro adyacente a la clase Hotel que es el todo y la habitación es la parte en la relación de composición).

La clase Habitación modela una habitación concreta del hotel que cuenta con atributos como su número de habitación, su disponibilidad y precio por día. Además, cuenta con un constructor y métodos *get* y *set* para cada atributo. La clase Habitación está asociada con la clase Huésped por medio de una relación de asociación. Según la multiplicidad de esta relación, una habitación puede ser ocupada por un único huésped. Un huésped tiene como atributos su nombre, apellidos, documento de identidad, fecha de ingreso y fecha de salida. La clase Huésped posee su constructor y métodos *get* y *set* para cada atributo. También tiene un método para calcular la cantidad de días que se alojó el huésped en la habitación.

La clase VentanaPrincipal posee atributos privados para crear una ventana gráfica que cuenta con un contenedor de componentes gráficos (*Container*), una barra de menú (*JMenuBar*), una opción de menú (*JMenú*) y dos opciones de menú (*JMenuItem*). La clase VentanaPrincipal cuenta con un constructor y métodos para generar la ventana gráfica con sus componentes (inicio) y para gestionar los diferentes eventos surgidos al interactuar con esta ventana (*actionPerformed*).

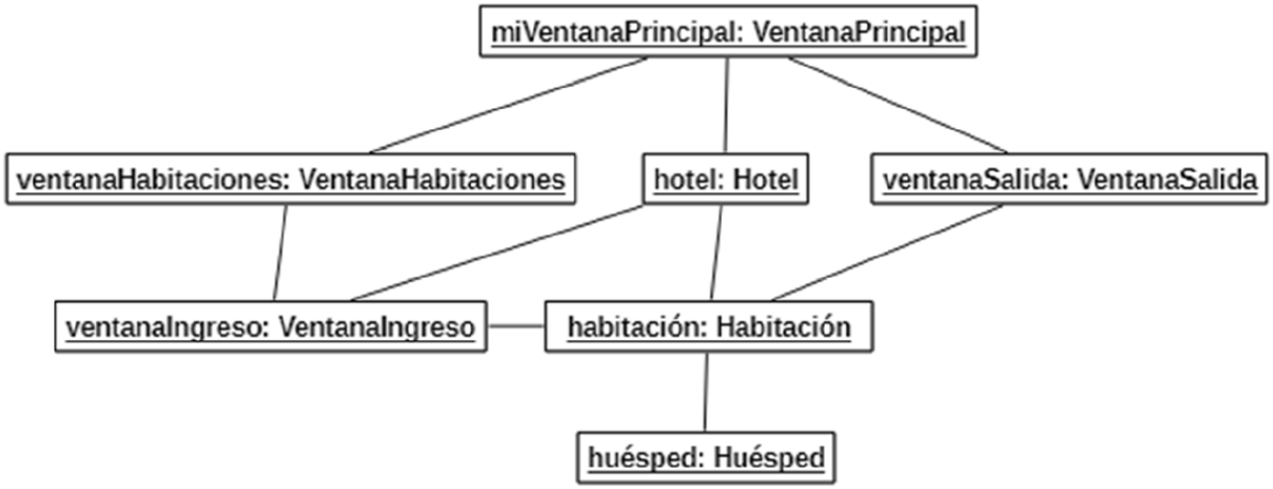
La clase VentanaPrincipal se relaciona con dos clases: VentanaHabitaciones y VentanaSalida, a través de los eventos generados al pulsar los dos ítems de menú. La clase VentanaHabitaciones genera una ventana gráfica

que muestra diez habitaciones junto con su disponibilidad. En esta ventana se debe ingresar el número de la habitación a ocupar. Para ello, la ventana define como atributos un contenedor de componentes gráficos (*Container*), diferentes etiquetas para identificar las habitaciones y su disponibilidad (*JLabel*), un selector numérico de la habitación a ocupar (*JSpinner* y *SpinnerNumberModel*) y un botón para aceptar el ingreso del número de habitación. La clase VentanaHabitaciones cuenta con su constructor y métodos para generar la ventana gráfica con sus componentes (inicio) y para gestionar la pulsación del botón (*actionPerformed*).

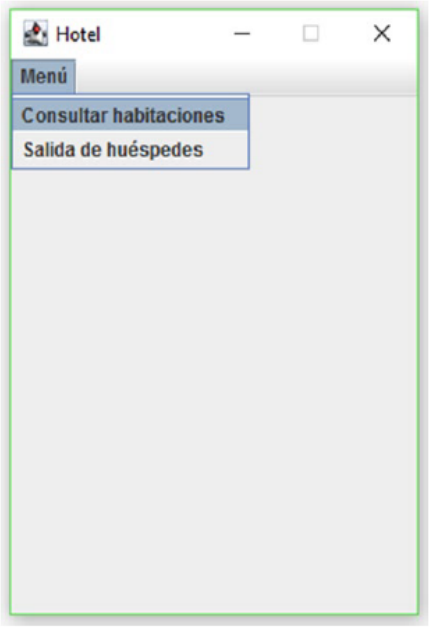
En primer lugar, la clase VentanaHabitaciones se vincula con la clase VentanaIngreso que permite ingresar los datos del huésped que ocupará la habitación. Para ello, la clase VentanaIngreso define como atributos un contenedor de componentes gráficos (*Container*), diferentes etiquetas para identificar la habitación y los campos a ingresar (*JLabel*), campos de texto para ingresar los datos del huésped (*JTextField*) y atributos para identificar la habitación a ocupar y la fecha de ingreso. Además, la clase VentanaHabitaciones posee un constructor y métodos para generar la ventana gráfica con sus componentes (inicio) y para gestionar la pulsación del botón (*actionPerformed*).

En segundo lugar, la clase VentanaSalida genera una ventana gráfica que permite registrar la entrega de una habitación y la salida de un huésped. Para ello, la clase cuenta con atributos como un contenedor de componentes gráficos (*Container*), diferentes etiquetas para identificar los campos a ingresar y calcular (*JLabel*), campos de texto para ingresar datos (*JTextField*) y botones para calcular el total a pagar, registrar la salida del huésped y liberar la habitación (*JButton*). Además, la clase VentanaSalida posee un constructor y métodos para generar la ventana gráfica con sus componentes (inicio) y gestionar la pulsación del botón de calcular pago y registrar salida (*actionPerformed*).

Diagrama de objetos

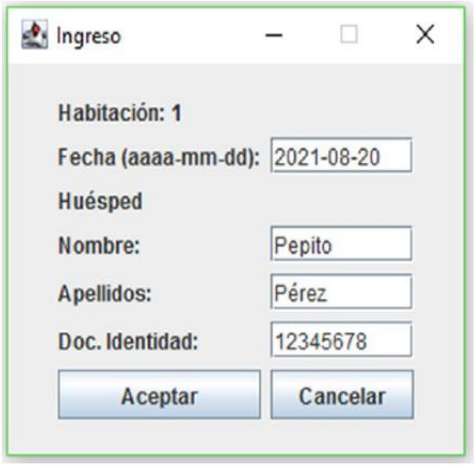


Ejecución del programa

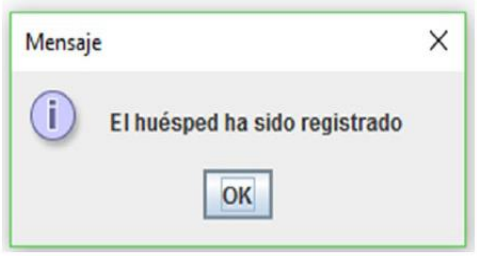


a) Menú principal

b) Selección de la habitación a ocupar

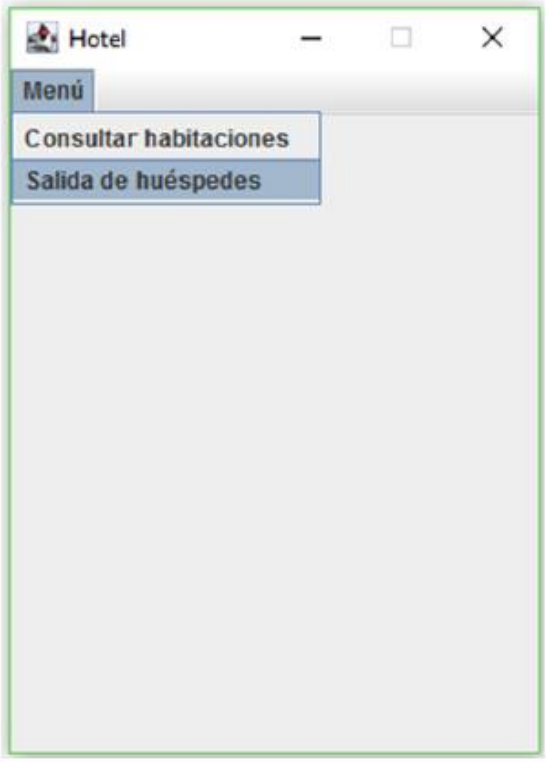


c) Ingreso del huésped

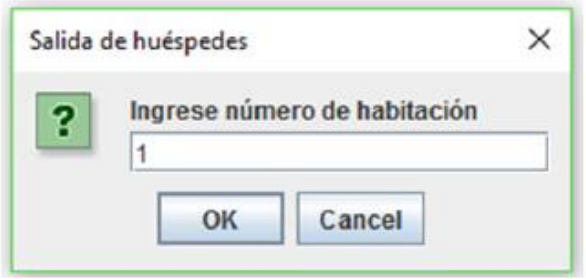


d) Confirmación del registro del huésped

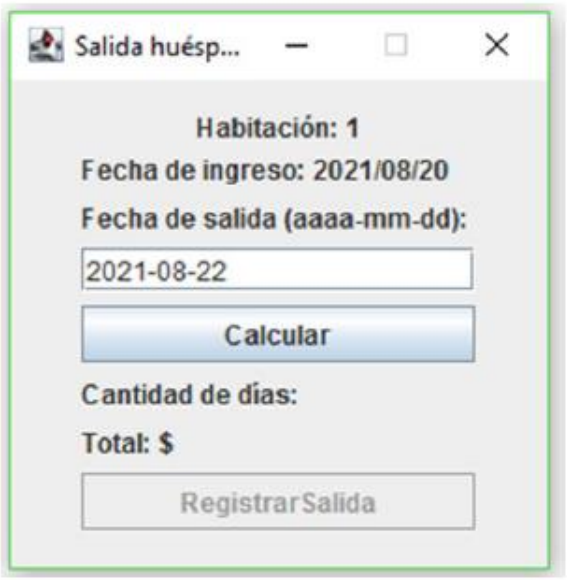
e) La habitación seleccionada aparece como "No disponible"



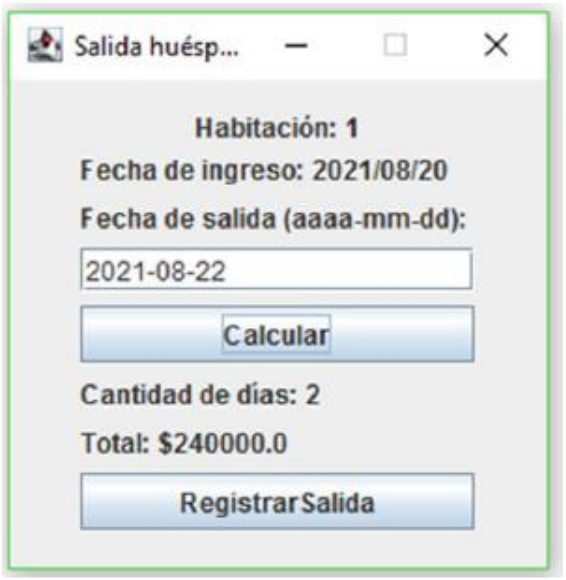
f) Menú salida de huésped



g) Ingresar número de habitación



h) Ingreso de la fecha de salida



i) Cálculo del total a pagar

Ejercicios propuestos

Modificar el programa del hotel para que soporte las siguientes funcionalidades:

- Las habitaciones pueden ser simples, dobles y triples. Cada tipo de habitación tiene un precio distinto. Cuando se registra el ingreso a una habitación se ingresan los datos de los huéspedes correspondientes.
- Se requiere de un informe histórico de los tiempos ocupados por cada habitación.
- Se requiere de un informe histórico de las habitaciones ocupadas por un determinado huésped.

Instrucciones

Clase	Método	Descripción
Integer	*Integer valueOf(int i)*	Retorna un objeto *Integer* que representa el valor *int* especificado.
String	*String format(String formato, Object … args)*	Retorna un *String* formateado utilizando el formato y argumentos especificados.
JFrame	*JFrame()*	Constructor de la clase *JFrame*.
	void setTitle(String titulo)	Establece el título de la ventana con el *String* especificado.
	void setSize(int x, int y)	Cambia el tamaño del componente para que tenga una anchura *x* y una altura *y*.
	void setLocationRelativeTo (Component c)	Establece la ubicación de la ventana en relación con el componente especificado.
	void setDefaultCloseOperation (opciones)	Usado para especificar una de las siguientes opciones del botón de cierre: EXIT_ON_CLOSE, HIDE_ON_CLOSE, DISPOSE_ON_CLOSE o DO_ NOTHING_ON_CLOSE.
	void setResizable(boolean resizable)	Para evitar que se cambie el tamaño de la ventana.
	void setVisible(boolean b)	Muestra u oculta la ventana según el valor del parámetro b.
ActionListener	*void actionPerformed(actionEvent e)*	Se invoca cuando ocurre un evento.
Container	*Container getContentPane()*	Retorna el objeto *ContentPane* de la ventana.
	void setLayout(LayoutManager mgr)	Establece el *layout* de la ventana.
	Component add(Component comp)	Añade el componente especificado al final del contenedor.
Component	*void addActionListener(this)*	Añade un oyente de eventos al componente actual.
	void setBounds(int x, int y, int ancho, int alto)	Mueve y cambia el tamaño del componente.
JLabel	*JLabel()*	Constructor de la clase *JLabel*.
	void setText(String text)	Define una línea de texto que mostrará este componente.

Clase	Método	Descripción
JTextField	*JTextField()*	Constructor de la clase *JTextField*.
	String getText()	Retorna el texto contenido en el componente de texto.
JButton	*JButton()*	Constructor de la clase *JButton*.
	void setText(String text)	Define una línea de texto que mostrará este componente.
	int getSelectedIndex()	Devuelve el índice seleccionado.
JMenuBar	*JMenuBar()*	Constructor de la clase *JMenuBar*.
	JMenu add(JMenu c)	Añade el menú especificado al final de la barra de menú.
JMenu	*JMenu()*	Constructor de la clase *JMenu*.
	JMenuItem add(JMenuItem menuItem)	Añade un ítem de menú al final del menú.
JMenuItem	*JMenuItem()*	Constructor de la clase *JMenuItem*.
JSpinner	*void setModel(SpinnerModel modelo)*	Cambia el modelo que representa el valor del *spinner*.
SpinnerNumberModel	*void setMinimum(Comparable mínimo)*	Cambia el mínimo inferior en la secuencia.
	void setMaximum(Comparable máximo)	Cambia el máximo superior en la secuencia.
	void setValue(Object valor)	Establece el valor actual en esta secuencia.
GridBagLayout	*GridBagLayout()*	Constructor de la clase *GridBagLayout*.
GridBagConstraints	*GridBagConstraints()*	Constructor de la clase *GridBagConstraints*.
	int HORIZONTAL	Cambie el tamaño del componente horizontalmente pero no verticalmente.
	int gridx	Especifica la celda inicial en el eje *x* del área de visualización del componente.
	int gridy	Especifica la celda inicial en el eje *y* del área de visualización del componente.

Insets	*Insets(int superior, int izquierda, int inferior, int derecha)*	Crea e inicializa un objeto *Insets* con las inserciones superior, izquierda, inferior y derecha especificadas.
SimpleDateFormat	*SimpleDateFormat(String patrón)*	Construye un *SimpleDateFormat* utilizando el patrón dado y los símbolos de formato de fecha predeterminados.
Date	*Date parse(Strings)*	Convierte el *String* a tipo *date*.
	int compareTo(Date otraFecha)	Compara dos fechas según orden.
	long getTime()	Retorna el número de milisegundos desde enero 1 de 1970 00:00:00.

Clase	Método	Descripción
Event	*Object getSource()*	El objeto sobre el cual el evento inicialmente ha ocurrido.
JOptionPane	*void showMessageDialog (Component componentePadre, Object mensaje)*	Crea un cuadro de diálogo.
	void showInputDialog (Component componentePadre, Object mensaje)	Muestra un cuadro de diálogo que solicita entrada de datos.