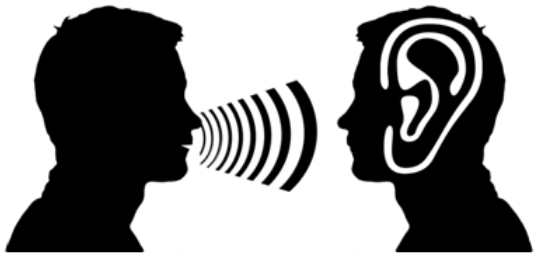
# 11.B. Eventos.

## 1. Eventos.

### 1.6. Creación de controladores de eventos.

A partir del JDK1.4 se introdujo en Java la clase `EventHandler` para soportar oyentes de evento muy sencillos.



La utilidad de estos controladores o manejadores de evento es:

- Crear oyentes de evento sin tener que incluirlos en una clase propia.

- Esto aumente el rendimiento, ya que no "añade" otra clase.

Como inconveniente, destaca la dificultad de construcción: **los errores no se detectan en tiempo de compilación**, sino en tiempo de ejecución.
Por esta razón, es mejor crear controladores de evento con la ayuda de un asistente y documentarlos todo lo posible.

El uso más sencillo de `EventHandler` consiste en instalar un oyente que llama a un método, en el objeto objetivo sin argumentos. En el siguiente ejemplo creamos un `ActionListener` que invoca al método *dibujar* en una instancia de `javax.Swing.JFrame`.

```
miBoton.addActionListener(
        (ActionListener)EventHandler.create(ActionListener.class, frame, "dibujar"));
```

Cuando se pulse miBoton, se ejecutará la sentencia `frame.dibujar()`. Se obtendría el mismo efecto, con mayor seguridad en tiempo de compilación, definiendo una nueva implementación al interface `ActionListener` y añadiendo una instancia de ello al botón:

```
// Código equivalente empleando una clase interna en lugar de EventHandler.
miBoton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                frame.dibujar();
        }
});
```

Probablemente el uso más típico de `EventHandler` es extraer el valor de una propiedad de la fuente del objeto evento y establecer este valor como el valor de una propiedad del objeto destino. En el siguiente ejemplo se crea un `ActionListener` que establece la propiedad "`label`" del objeto destino al valor de la propiedad "`text`" de la fuente (el valor de la propiedad "`source`") del evento.

```
EventHandler.create(ActionListener.class, miBoton, "label", "source.text")
```

Esto correspondería a la implementación de la siguiente clase interna:

```
// Código equivalente utilizando una clase interna en vez de EventHandler.
new ActionListener {
        public void actionPerformed(ActionEvent e) {
```

```
            miBoton.setLabel(((JTextField)e.getSource()).getText());
        }
}
```

## Autoevaluación

**El uso de** EventHandler **tiene como inconveniente, que los errores no se detectan en tiempo de ejecución.**

- ○ Verdadero.
- ○ Falso.