Para que este patrón funcione, la clase base de diálogo debe funcionar con botones abstractos, es decir, una clase base o una interfaz que sigan todos los botones concretos. De este modo, el código sigue siendo funcional, independientemente del tipo de botones con el que trabaje.

Por supuesto, también se puede aplicar este sistema a otros elementos UI. Sin embargo, con cada nuevo método de fábrica que añadas al diálogo, más te acercarás al patrón **Abstract Factory**. No temas, más adelante hablaremos sobre este patrón.

```
// La clase creadora declara el método fábrica que debe devolver
// un objeto de una clase de producto. Normalmente, las
// subclases de la creadora proporcionan la implementación de
// este método.
class Dialog is
  // La creadora también puede proporcionar cierta
  // implementación por defecto del método fábrica.
  abstract method createButton():Button

  // Observa que, a pesar de su nombre, la principal
  // responsabilidad de la creadora no es crear productos.
  // Normalmente contiene cierta lógica de negocio que depende
  // de los objetos de producto devueltos por el método
  // fábrica. Las subclases pueden cambiar indirectamente esa
  // lógica de negocio sobrescribiendo el método fábrica y
  // devolviendo desde él un tipo diferente de producto.
  method render() is
    // Invoca el método fábrica para crear un objeto de
    // producto.
```

```
    Button okButton = createButton()
    // Ahora utiliza el producto.
    okButton.onClick(closeDialog)
    okButton.render()


// Los creadores concretos sobrescriben el método fábrica para
// cambiar el tipo de producto resultante.
class WindowsDialog extends Dialog is
  method createButton():Button is
    return new WindowsButton()

class WebDialog extends Dialog is
  method createButton():Button is
    return new HTMLButton()


// La interfaz de producto declara las operaciones que todos los
// productos concretos deben implementar.
interface Button is
  method render()
  method onClick(f)

// Los productos concretos proporcionan varias implementaciones
// de la interfaz de producto.

class WindowsButton implements Button is
  method render(a, b) is
    // Representa un botón en estilo Windows.
  method onClick(f) is
    // Vincula un evento clic de OS nativo.

```

```
class HTMLButton implements Button is
  method render(a, b) is
    // Devuelve una representación HTML de un botón.
  method onClick(f) is
    // Vincula un evento clic de navegador web.

class Application is
  field dialog: Dialog

  // La aplicación elige un tipo de creador dependiendo de la
  // configuración actual o los ajustes del entorno.
  method initialize() is
    config = readApplicationConfigFile()

    if (config.OS == "Windows") then
      dialog = new WindowsDialog()
    else if (config.OS == "Web") then
      dialog = new WebDialog()
    else
      throw new Exception("Error! Unknown operating system.")

  // El código cliente funciona con una instancia de un
  // creador concreto, aunque a través de su interfaz base.
  // Siempre y cuando el cliente siga funcionando con el
  // creador a través de la interfaz base, puedes pasarle
  // cualquier subclase del creador.
  method main() is
    this.initialize()
    dialog.render()
```