1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1

</ Clase 3 - POO

- Mensajes más comunes.
- Especificación de clase simple.
- Desarrollo de una aplicación utilizando la especificación de una clase.
- Implementación de clase simple.
- Desarrollar Aplicación en Dolphin.
- Ejercicios para resolver.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Repaso

Tipos de Mensajes

Orden de los mensajes

Componentes de un mensaje

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

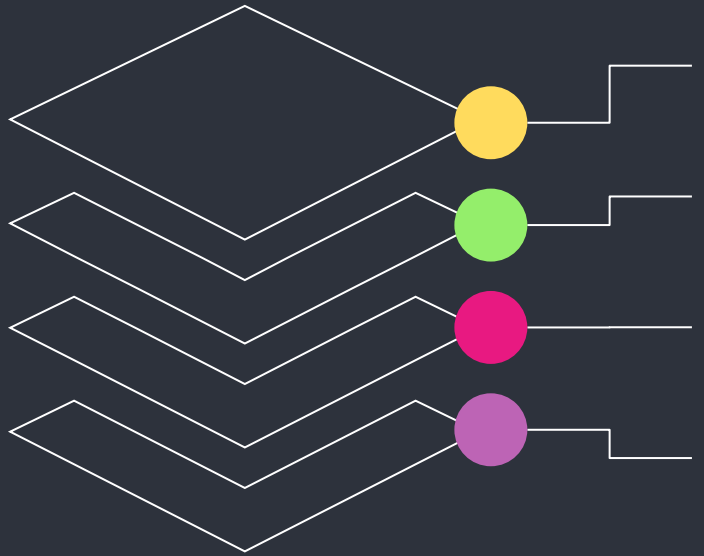
</>

Mensajes más comunes

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Clase Transcript



{show} Muestra el string en el Transcript

{cr} Salto de Línea

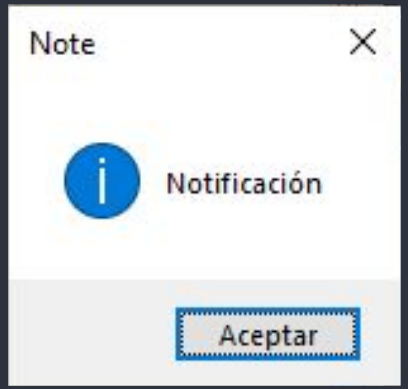
{space} Deja un espacio

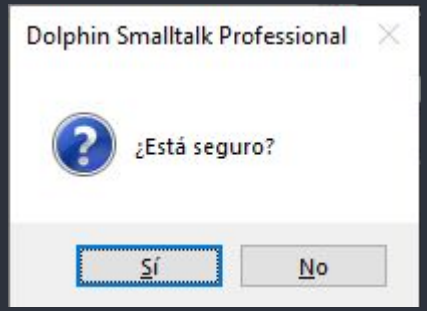
{clear} Limpia el Transcript

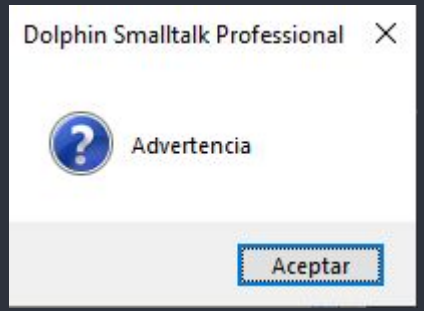
1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1

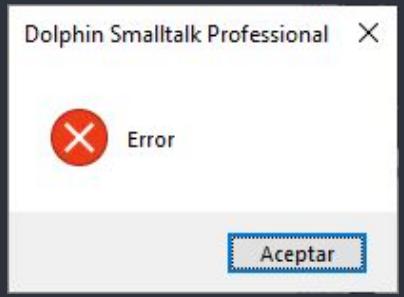
</Clase MessageBox

notify	confirm (retorna bool)	warning	errorMsg





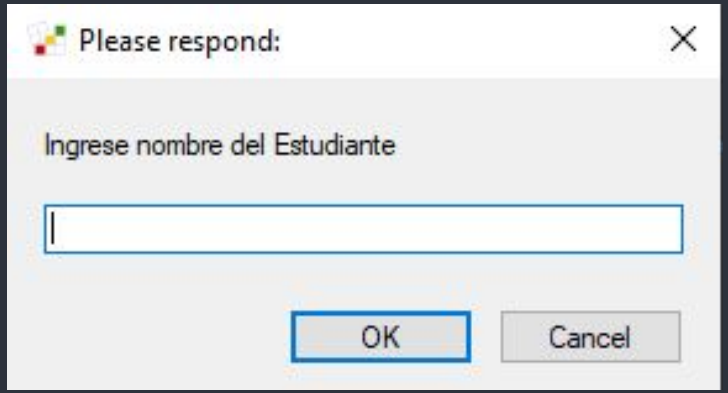




1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</Clase Prompter

Prompt: permite que el usuario ingrese información por teclado:



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</Otros mensajes

- {Inspect} Muestra información detallada del objeto a debuguear
- {Size} Devuelve el tamaño de un arreglo
- {^} Retorna un valor
- {class} Retorna la clase de un objeto
- {superclass} Retorna la super clase de un objeto
- {notNill} Indica si el objeto es nil o no

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1

</>

Especificación clase simple

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Especificación clase Remedio

Definición:

Object subclass: #Remedio

instanceVariableNames: ‘nombre precio stock laborat’

classVariableNames:''

PoolDictionaries:''

Métodos de clase

crear:nom con:unPre con:unSt con:unLab

“Crea una instancia de la clase Remedio y la inicializa con sus datos"

</Especificación clase Remedio

Métodos de Instancia:

>>iniciar: unNom pre:unPre stock:unStock laborat:unLab
"Inicializa el remedio con sus datos"

>>verNombre *"Retorna el nombre del remedio"*

>>verStock *"Retorna el stock del remedio"*

>>verPrecio *"Retorna el precio del remedio"*

>>verLab *"Retorna el laboratorio del remedio"*

>>modNombre: otroNom *"Modifica el nombre del remedio"*

>>modStock: otroSt *"Modifica el stock del remedio"*

>>modPrecio: otroPre *"Modifica el precio del remedio"*

>>modLab: otroLab *"Modifica el laboratorio del remedio"*

</>

Desarrollar Aplicación

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Enunciado

Teniendo en cuenta la especificación de la clase remedio, escribir una aplicación que permita crear y cargar dos instancias de la clase. Luego:

+ Retornar el nombre y el stock del remedio más económico.
+ Realizar un incremento del 20% al remedio con menor stock. Mostrar dicho cambio en el Transcript.

</>

Implementación clase simple

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Implementación clase Remedio

Definición:

Object subclass: #Remedio

instanceVariableNames: ‘nombre precio stock laborat’

classVariableNames:''

PoolDictionaries:''

Métodos de clase

crear:nom con:unPre con:unSt con:unLab

“Crea una instancia de la clase Remedio y la inicializa con sus datos"

^self new iniciar: unNom pre: unPre stock: unStock laborat: unLab

</Implementación clase Remedio

Métodos de Instancia: Inicializar

```
>>iniciar: unNom pre:unPre stock:unStock laborat:unLab
"Inicializa el remedio con sus datos"
    nombre:=unNom.
    precio:=unPre.
    stock:=unStock.
    laborat:=unLab.
```

</Implementación clase Remedio

Métodos de Instancia: ver atributos

```
>>verNombre "Retorna el nombre del remedio"
    ^nombre
>>verStock "Retorna el stock del remedio"
    ^precio
>>verPrecio "Retorna el precio del remedio"
    ^stock
>>verLab "Retorna el laboratorio del remedio"
    ^laborat
```

</Implementación clase Remedio

Métodos de Instancia: modificar atributos

```
>>modNombre: otroNom "Modifica el nombre del remedio"
    nombre:=otroNom.
>>modStock: otroSt "Modifica el stock del remedio"
    precio:=otroPre.
>>modPrecio: otroPre "Modifica el precio del remedio"
    stock:=otroStock.
>>modLab: otroLab "Modifica el laboratorio del remedio"
    laborat:otroLab.
```

</>

Ejercicios para resolver

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Enunciado

Teniendo en cuenta la especificación de la clase Robot (siguiente filmina), desarrolle la siguiente aplicación:

Suponer que se tiene una grilla que representa una ciudad de 50*50 cuadras. Desarrollar una aplicación que permita crear un robot llamado ‘Rolo’, posicionarlo en la esquina (20,30) y hacer que recorra un rectángulo de 5 cuadras de altura por 10 cuadras de ancho.

</Especificación clase Robot

Método(s) de clase:

>>crearRobot: unNombre

"Retorna al robot posicionado en la esquina (1,1), con orientación hacia el norte, con el nombre 'unNombre'"

>>mover *"Posiciona al robot en la esquina siguiente sin modificar su orientación"*

>>derecha *"rota 90° a la derecha al robot"*

>>izquierda *"rota 90° a la izquierda al robot"*

>>posx "retorna la coordenada x de la esquina donde está el robot"

>>posy "retorna la coordenada y de la esquina donde está el robot"

>>posx:unX posy:unY "Posiciona al robot en la esquina (x,y)"