

## REPASANDO MENSAJE Y MÉTODO

Mensaje y método no es sólo un juego de palabras.

Me define de qué lado me pongo:

- **Mensaje:** soy usuario del objeto. Soy un observador que perturbo al objeto (perturbo en un sentido más físico que molesto).
- **Método:** yo soy un objeto, que ejecuto código que está **dentro** mío, más allá de quién lo haya pedido.

## INTERFAZ. IMPLEMENTACIÓN.

También podemos separar un objeto entre lo que es: interfaz e implementación.

- **Interfaz** es lo que un objeto publica para que otro objeto lo use (es el *qué*).
- **Implementación** es lo que un objeto encapsula para definir *cómo* se termina resolviendo un mensaje.

¿Para qué me sirve esto?

Cuando era chico tenía un grabador y jugaba a hacer una radio. Así aprendí las funciones básicas de un grabador de audio:



Botón "play": me permitía escuchar lo que tuviera el cassette

Años más tarde aparecieron los reproductores y las videocassetteras de VHS:



Nuevamente el botón play, con la misma interfaz: ▶

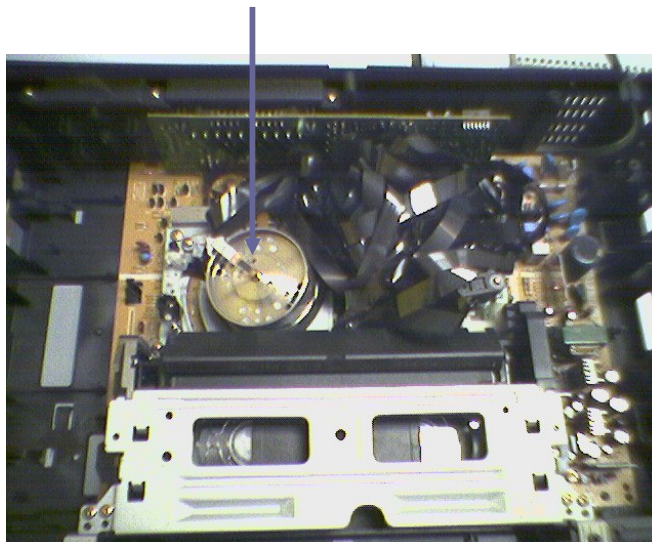
¿Y qué pasa cuando tenemos un reproductor de DVD?



¡la misma interfaz! Con play podemos ver lo que tiene ese soporte

¿Qué ventaja trae usar siempre la misma interfaz?

El usuario no conoce las implementaciones internas de cada aparato:



Sólo se concentra en decir qué mensaje tiene que enviar.

## POLIMORFISMO

Otra ventaja (y una muuuuuuy importante): es el **polimorfismo: a todos los objetos les hablo igual, sólo que cada uno responde a su manera.**

El polimorfismo no es para la videocassettera, ni para la lectora de DVD, ni para el radiograbador, **sino que es para el que los usa. Siempre** es para el que usa los objetos polimórficos. ¿Qué tiene de bueno? Que se puede hacer algo como:

```
divertiteCon: unObjeto
    unObjeto play.
```

Y no tengo que saber si unObjeto es un dvd, un cassette de audio o de video, o algo que todavía no se inventó.

El polimorfismo es un concepto parecido a la belleza. Jennifer Connely en una isla desierta no es ni linda ni fea, **es**. Ahora, una persona es bella en tanto hay alguien que la observa y la juzga bella<sup>1</sup>.



De la misma manera, un objeto no sabe si es polimórfico, ni tiene idea del contexto donde se lo utiliza. Sólo sabe responder una serie de cosas que forman parte de sus responsabilidades. El observador/el cliente/el que usa el objeto es el que dice: "Mmmm... un triángulo, un cuadrado y un círculo tienen área, yo abstraigo la idea de figura y le pido que me de su área. Cada uno sabrá cómo calcularla"

¿Cuántos objetos necesito para tener polimorfismo?

3:

---

<sup>1</sup> Esta noción fue adaptada de una idea de Alejandro Dolina

1 observador y 2 objetos que puedan intercambiarse sin que el observador note la diferencia.

Tengo que calcular el monto que le voy a pagar a un empleado:

- Los empleados en relación de dependencia cobran un sueldo fijo
- Los vendedores cobran una comisión del total de las ventas que tuvieron en el mes.

¿Qué me conviene hacer? En pseudocódigo smalltalkero:

<pre> Función calcularSueldo() {  Si el tipo de empleado = 73 "Relación de dependencia"     ↑ empleado sueldoFijo  Si el tipo de empleado = 82 "Vendedor"     ↑ empleado totalVentas * empleado comisión  }         </pre>	<p>↑ <i>empleado</i><sup>1</sup> sueldo</p> <p>En cada objeto tendré la lógica dependiendo de si es vendedor o empleado en relación de dependencia.</p> <p>Para el empleado gimenez sueldo ↑ sueldoFijo</p> <p>Para el vendedor gonzalez sueldo ↑ totalVentas * comision</p> <p>(1) Ese empleado es una variable: puede referenciar a gimenez o gonzalez.</p>
--	---

La opción 2) **desacopla** al observador del empleado que tiene en ese momento. Sólo debe conocer una cosa: el mensaje que le tiene que mandar (la interfaz del objeto: sueldo). El nombre del método es igual en los dos objetos para que el que invoca no tenga que preguntar por el tipo de objeto. Si en una solución con objetos estamos preguntando por el tipo, algo no estamos haciendo bien.

### CONOCIENDO AL SMALLTALK

Smalltalk es la implementación que vamos a usar dentro del paradigma orientado a objetos.

¿Cómo se instala?

[www.pdep.com.ar](http://www.pdep.com.ar)

Software > Pharo

(otra alternativa es instalar Dolphin Smalltalk, pero como está discontinuado vamos a preferir Pharo).

En Smalltalk el ambiente se llama imagen, y es un archivo visible desde el Explorador de Windows.

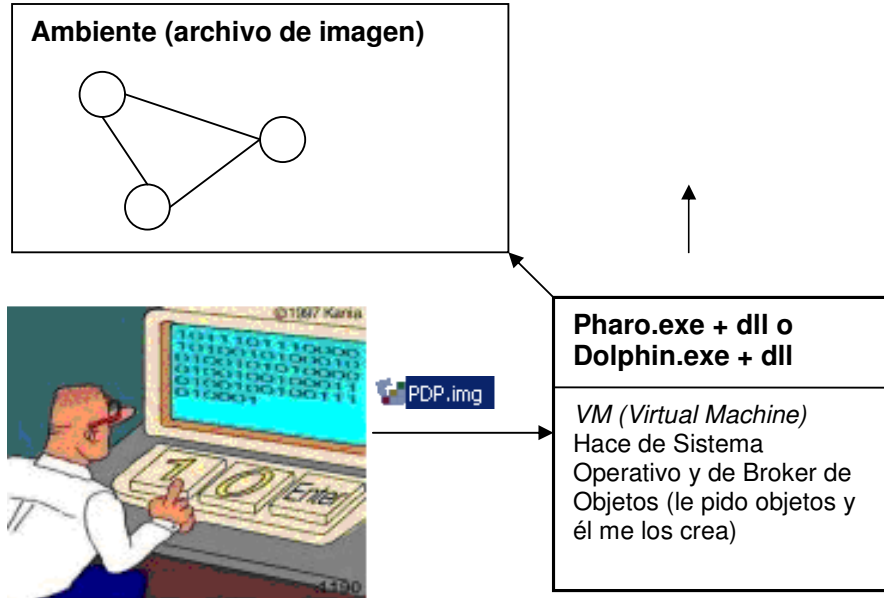
En Pharo:

- Pharo-pdep-1.1.image

En Dolphin Smalltalk<sup>2</sup>:

- PDP.img

<sup>2</sup> En el ejemplo renombramos la imagen como nos aconseja el instructivo. El nombre que elegimos es PDP.img y el mismo nombre PDP tendrán los archivos de la página siguiente.



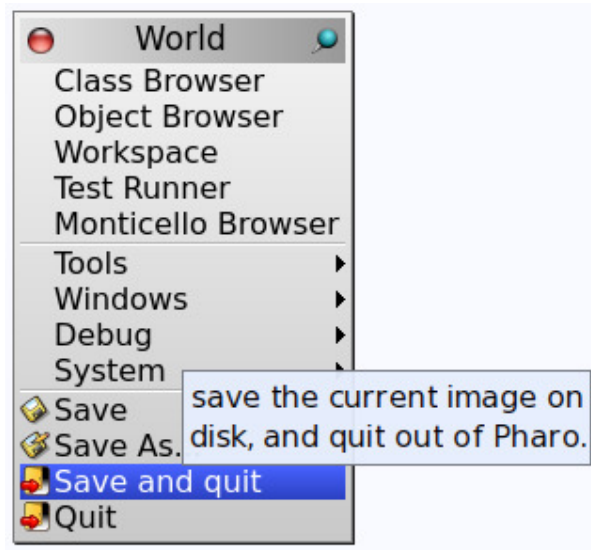
### ***COSAS IMPORTANTES:***


Toda imagen tiene un archivo de log de modificaciones a la imagen inicial (incluye código escrito). En caso de haber perdido la imagen original puede servir como último recurso de recuperación de fuentes

En Pharo ese archivo es Pharo-pdep-1.1.changes (si bajamos el rar de la cátedra).  
 En Dolphin se llama PDP.chg (o bien <nombre de la imagen>.chg)

Una buena práctica consiste en actualizar de tanto en tanto la imagen (donde están tanto los objetos que vamos creando como el código que se va desarrollando).

En Pharo, con un click izquierdo sobre el fondo (en la parte blanca) y luego Save / Save and Quit:



En Dolphin: haciendo click sobre el ícono correspondiente:  o desde el menú File → Save Image.

Cuando abrimos el Smalltalk, hay una Máquina Virtual o Ambiente que toma el control sobre una imagen: allí están todos mis objetos (mi desarrollo y mi ejecutable viven en el mismo ambiente, todos son objetos, esto trae consecuencias que vamos a analizar más tarde).

¿Qué tengo que hacer ahora? Leer los primeros dos capítulos del apunte de Objetos de la cátedra:

<http://www.pdep.com.ar>

Material > Apuntes > [apunte-objetos-1-4.doc](#)