doSomethingLater

```
[ self doSomething ] valueAfterWaiting: 3000 milliSeconds asDelay
```

Closures

Clausuras léxicas

```
doSomethingLater() {
    let arrowFunction = () => {this.doSomething()};
    window.setTimeout(arrowFunction, 3000);
}
```

Closures / clausuras léxicas

- Los closures (o clausuras léxicas) nos permiten diferir la evaluación de expresiones
- Situaciones frecuentes comunes en los que utilizo clausuras son:
 - Para indicar como continuar cuando esté listo el resultado de una tarea que se ejecuta de manera asincrónica (callbacks)
 - Cuando quiero tener funciones, que toman otras funciones como parámetro o que retornan funciones que luego voy a utilizar (por ejemplo, iteración)
 - En general, cuando quiero que sea posible cambiar (en parte) el comportamiento de un objeto, sin necesidad de su clasificar o modificar el código existente
 - Muy útil en la construcción de librerías y frameworks

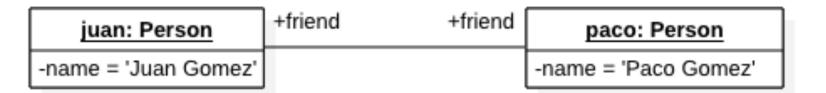
BlockClosure / Bloques (Repaso)

- En Smalltalk, los bloques (p.e., [Transcript show: 'hola']) son clausuras
- Una expresión entre corchetes, crea un bloque.
- Puede hacer referencia a cualquier variable disponible en el contexto en que se define (por ejemplo, variables de instancia, variables temporales de un método, parámetros de un método)
- Puedo asignarlo a variables y pasarlo como parámetro
- Puede hacer referencia a parámetros que deberá recibir al momento de ser evaluado
- Se evalúa enviando variantes del mensaje #value

En el playground ...

```
nombre := 'Juan Gomez'.
aBlockClosure := [ Transcript show: 'Hola ', nombre; cr ].
aBlockClosure value.
button := PluggableButtonMorph new .
button label: 'Click me'.
button position: 400@10.
button actionBlock: aBlockClosure.
button openInWorld .
```

En métodos ...



blockClosure

```
^ [ Transcript show: 'Hola, yo soy ' , self name ]
```

avaluateYourBlock

self blockClosure value

avaluateYourFriendsBlock

self friend blockClosure value

```
juan := Person named: 'Juan Gomez'.
paco := Person named: 'Paco Gomez'.
juan friend: paco.
juan evaluateYourBlock.
juan evaluateYourFriendsBlock.
```

Closures / clausuras léxicas

- Una clausura es una expresión que se define en el contexto de otra expresión
- En la clausura se puede hacer referencia a elementos definidos en su contexto (por ejemplo, variables).
- La pseudovariable "self" corresponde al contexto en el que se define la clausura
- La clausura "se lleva" las variables del contexto en el que fué definida. Si cambian "afuera" afecta a la clausura. Si se modifican en la clausura, tiene efecto "afuera".

Closures en otros lenguajes

- En Smalltalk los bloques son Closures
 - El "if", "while", y los iteradores se basan en ellos
 - Los procesos se crean a partir de closures (con el mensaje #fork)
 - Muchos objetos reutilizables se pueden configurar con bloques
- En Javascript, las funciones son closures (si se usa la notación de flecha, la pseudovariable this se linkea al this del contexto)
 - Se utilizan mucho, en especial para hacer callbacks (asincronismo)
- En Java, las expresiones lambda y las inner clases nos permiten implementar closures (ojo con la pseudovariable this)
- En otros lenguages (Pascal, PHP) no tenemos closures

Cuidados

- Los bloques son muy potentes, pero no debemos abusar
- Complican la depuración y hacen los programas mas difíciles de entender (en especial si no cuento con buenas herramientas)
- Una vez creados, no se pueden modificar (hay que volver a crearlos)
- No utilizo bloques cuando lo mismo puedo conseguirlo sin ellos