Programación III

Clase VII

Andrés Fortier

Importante

- Este sábado hay clase de Programación 3 a las 9:00.

Novedad

- La pregunta de la Rhodesia
  - (puede que el premio vaya cambiando con el tiempo).
- Pregunta al inicio de la clase sobre la clase anterior.
- Pregunta estilo final.
- Sólo una oportunidad de responder.
- Carpeta cerrada.
- Para ganar el premio la respuesta debe ser exacta.

La pregunta de la Rhodesia

- Enumere las cuatro formas de conocimiento entre objetos indicando con qué variables se asocian.

Repaso: Formas de conocimiento

- Cuatro formas de conocimiento entre objetos:
  - Conocimiento Interno: Variables de instancia.
  - Conocimiento Externo: Parámetros.
  - Conocimiento Temporal: Variables temporales.
  - Conocimiento Global: Variables globales.
- Además existe una quinta forma de conocimiento especial: las pseudo-variables
  - self.

Repaso: Formas de conocimiento

```
calcularCon: unNumero

    | temporal |

    temporal := velocidad + unNumero.
    ^temporal * self multiplicador.
```

Calculador
velocidad
multiplicador calcularCon: unNumero

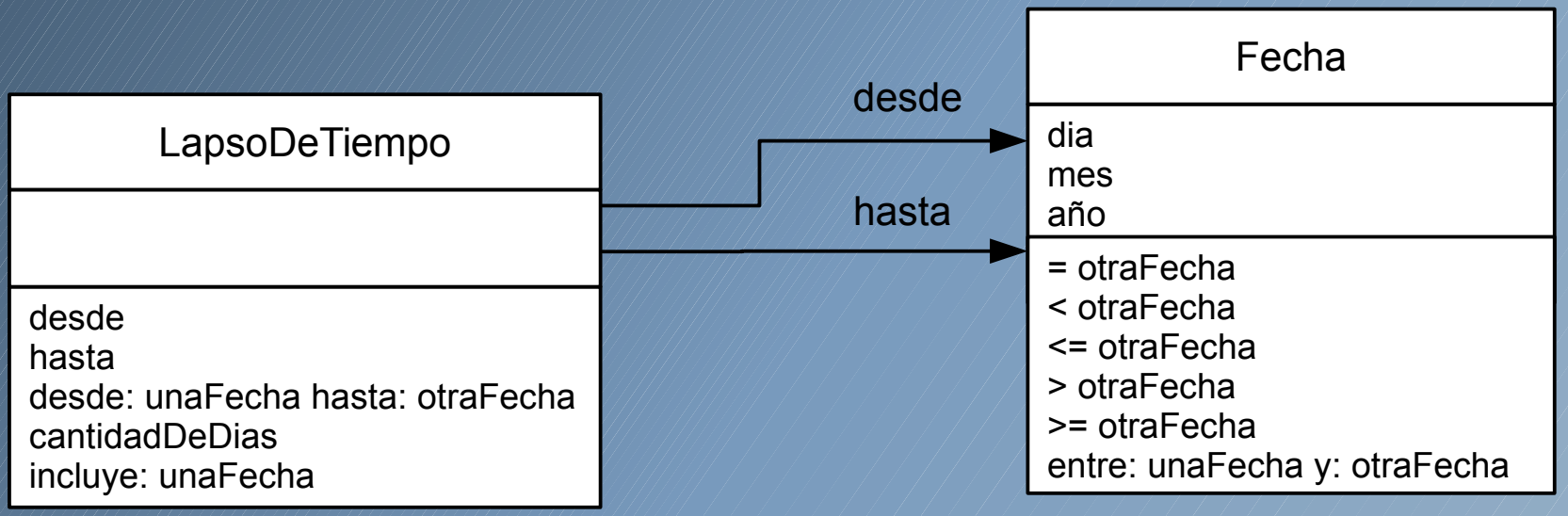
```
multiplicador

    ^ValorGlobal
```

Repaso: Lapso de tiempo

- Diseñar e implementar un objeto lapso de tiempo.
- Protocolo
  - `#desde`
  - `#hasta`
  - `#desde: unaFecha hasta: otraFecha`
  - `#cantidadDeDias`
  - `#incluye: unaFecha`

Repaso: Lapso de tiempo




```
desde

    ^desde.
```

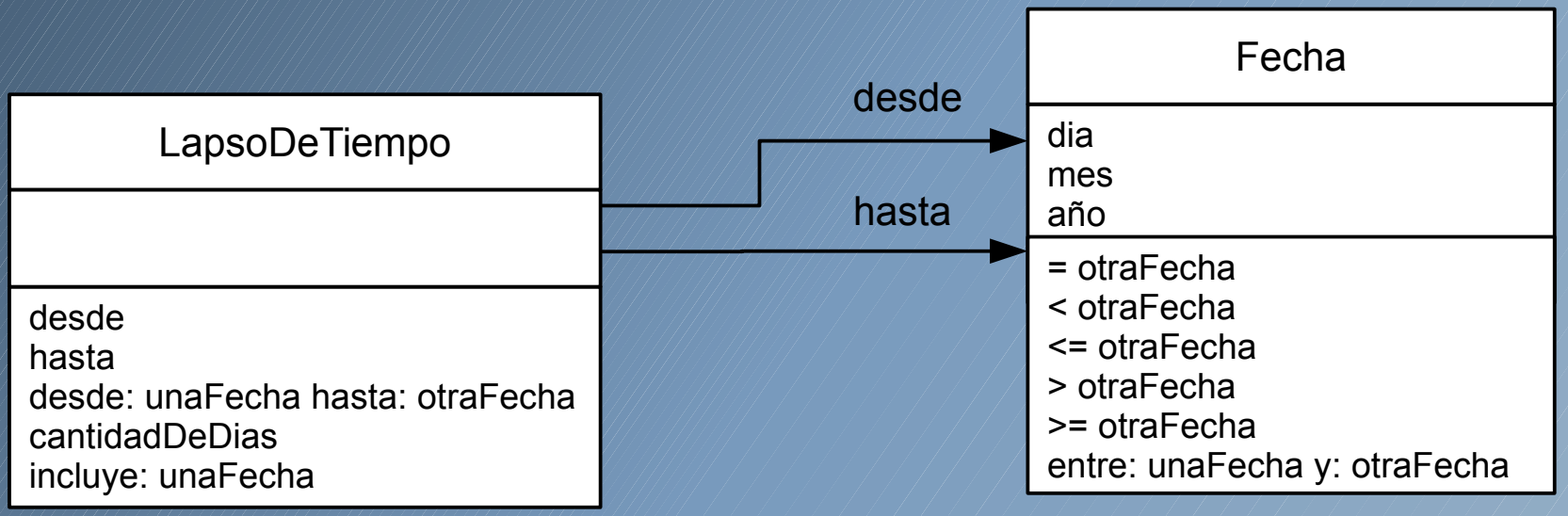
```
hasta

    ^hasta.
```

```
desde: unaFecha hasta: otraFecha

    desde := unaFecha.
    hasta := otraFecha.
```

Repaso: Lapso de tiempo




```
incluye: unaFecha

    ^unaFecha entre: self desde y: self hasta.
```

Inicialización

- Dijimos que un objeto existe para hacer algo.
- Una instancia recién creada …
  - ¿está lista para poder colaborar con otros objetos?
  - Pensemos en la creación de un objeto fecha.
  - O un lapso de tiempo o una cuenta bancaria.

Objetos “bien formados”

- Existe un tiempo entre:
  - La creación de un objeto.
  - La adquisición de los colaboradores que ese objeto necesita para llevar adelante sus responsabilidades.
- Mientras tanto, el objeto está en un estado “inconsistente”.
- ¿Cómo impedimos que esto suceda?
  - Inicializar al objeto con valores por defecto.
  - No permitir la creación de un objeto a menos que nos pasen la información mínima indispensable.

Constructores

- Son mensajes **de clase** relacionados a la construcción de objetos bien formados.
- Podemos crear constructores que tomen como parámetros los colaboradores básicos del nuevo objeto
  - Así, creamos objetos bien formados.
  - Un objeto bien formado se encuentra en estado consistente desde su creación.

Constructores

- Con estos constructores favorecemos
  - La legibilidad (la creación de objetos complejos se realiza en un solo paso).
  - La documentación (aquél que quiera crear nuevos objetos ya sabe cómo hacerlo).

Ejemplo

```
| fecha |

fecha := Fecha new.
fecha dia: unDia.
fecha mes: unMes.
fecha año: unAño.
^fecha.
```

Ejemplo

```
Fecha class >> dia: unDia mes: unMes año: unAño

    | fecha |

    fecha := self new.
    fecha dia: unDia.
    fecha mes: unMes.
    fecha año: unAño.
    ^fecha.
```

```
| hoy |

hoy := Fecha dia: 25 mes: 8 año: 2015.
hoy dia. "printIt => 25"
```

Más de un constructor

```
CuentaBancaria class >> titular: unaPersona saldo: unSaldo
    | cuenta |

    cuenta := self new.
    cuenta titular: unaPersona.
    cuenta saldo: unSaldo.
    ^cuenta.
```

```
CuentaBancaria class >> titular: unaPersona

    ^self titular: unaPersona saldo: 0.
```

```
| pepe juan cuentaPepe cuentaJuan |

pepe := …
juan := …
cuentaPepe := CuentaBancaria titular: pepe saldo: 1000.
cuentaJuan := CuentaBancaria titular: juan.
```

Constructores

- Podemos también redefinir el mensaje básico de construcción: #new.
- Y hacer que envíe un mensaje de inicialización a la instancia.

Ejemplo

```
Contador class >> new

    | nuevoContador |

    nuevoContador := super new.
    nuevoContador inicializar.
    ^nuevoContador.
```

```
| contador |

contador := Contador new.
contador valor. "printIt => 0"
```

New e initialize

- Pharo ya hace esto.
- Si definimos el método #initialize, al crear una nueva instancia se enviará este mensaje.

Ejemplo

```
Contador >> initialize

   valor := 0.
```

```
| contador |

contador := Contador new.
contador valor. "=> 0"
```