Programación III

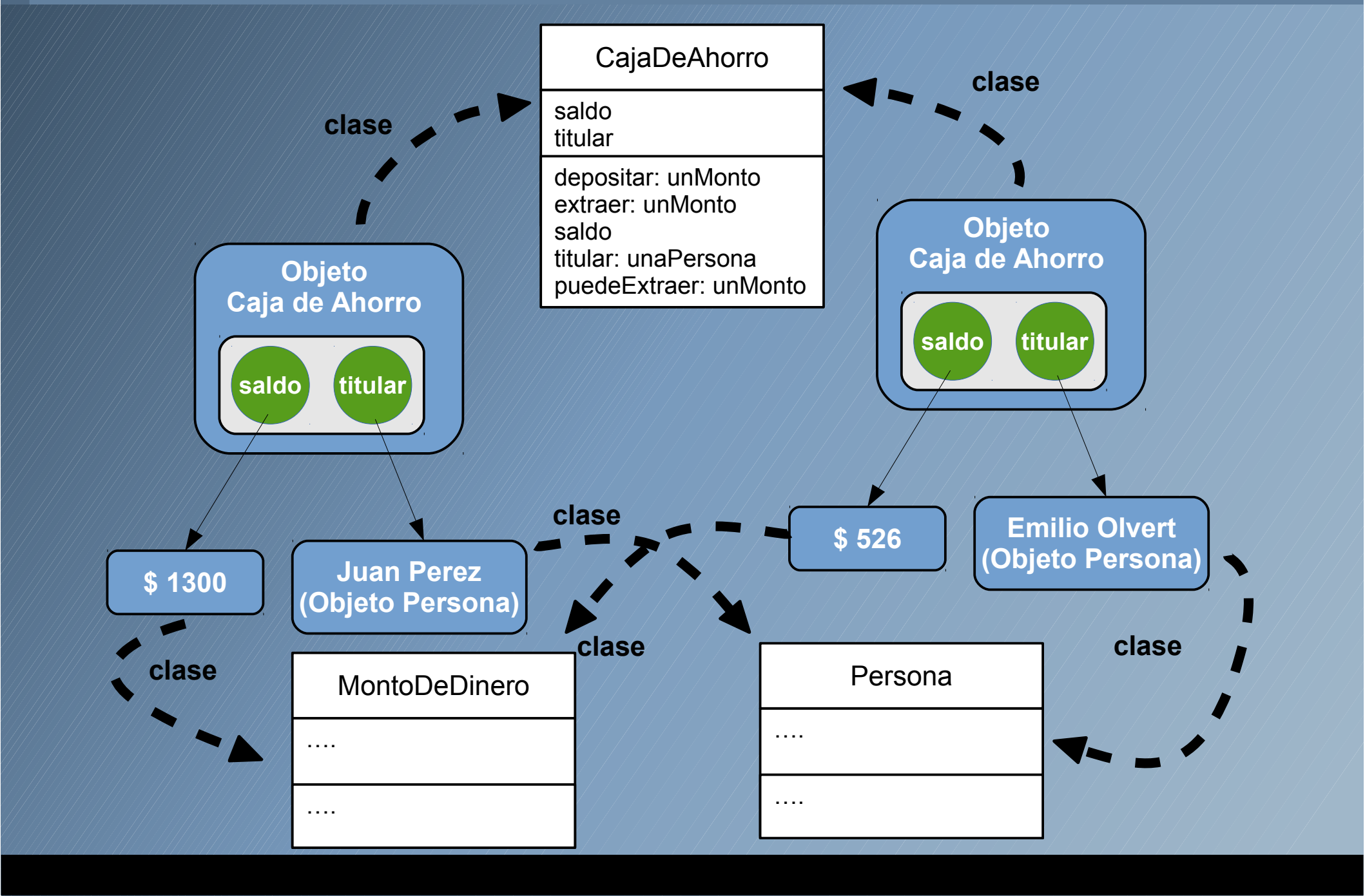
Clase VI

Andrés Fortier

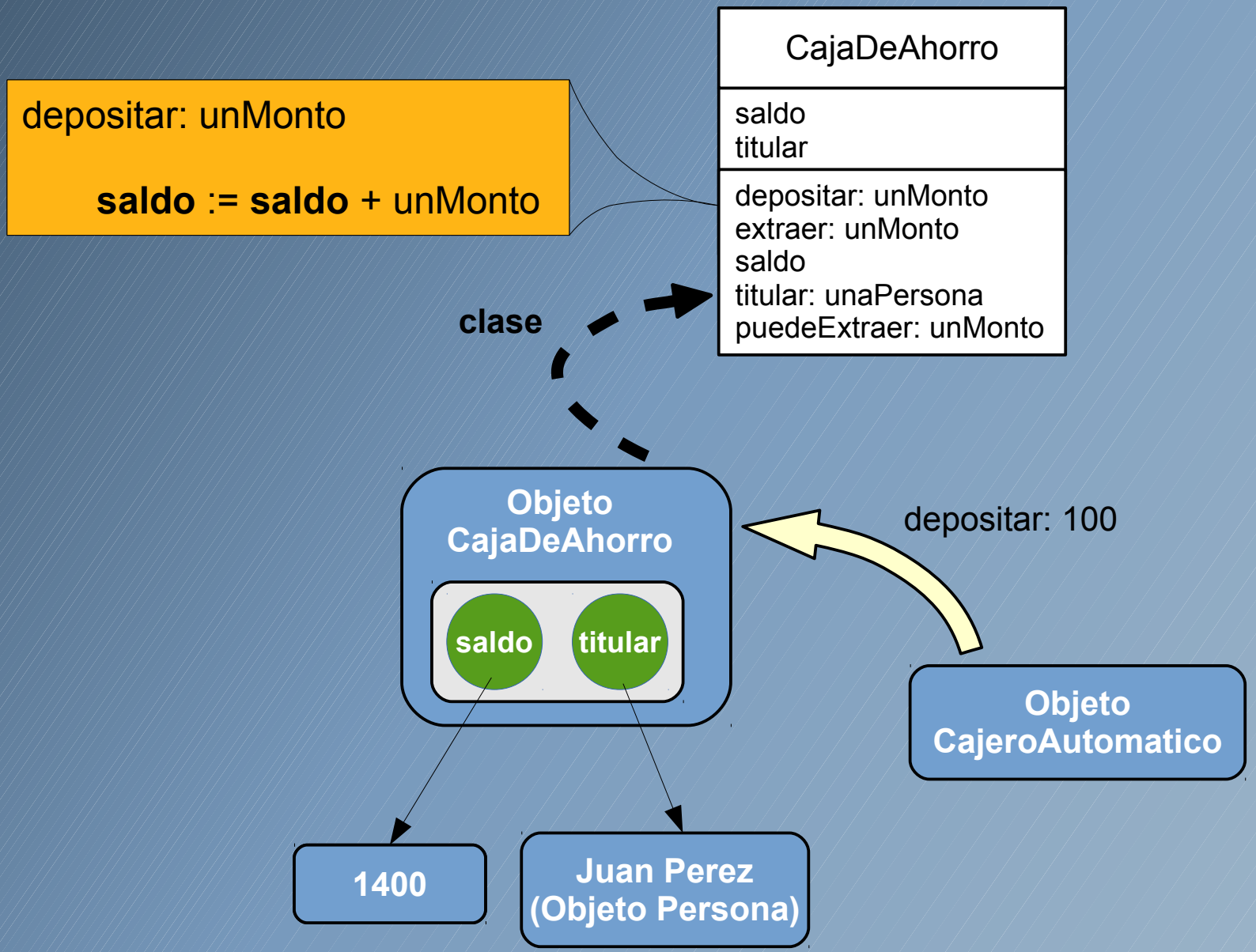
Repaso - Clases

- Una clase es una descripción abstracta de un conjunto de objetos.
- Las clases cumplen tres roles:
  - Agrupan el comportamiento común a sus instancias.
  - Definen la forma de sus instancias.
  - Crean objetos que son instancia de ellas.

Ejemplo de Clases e Instancias

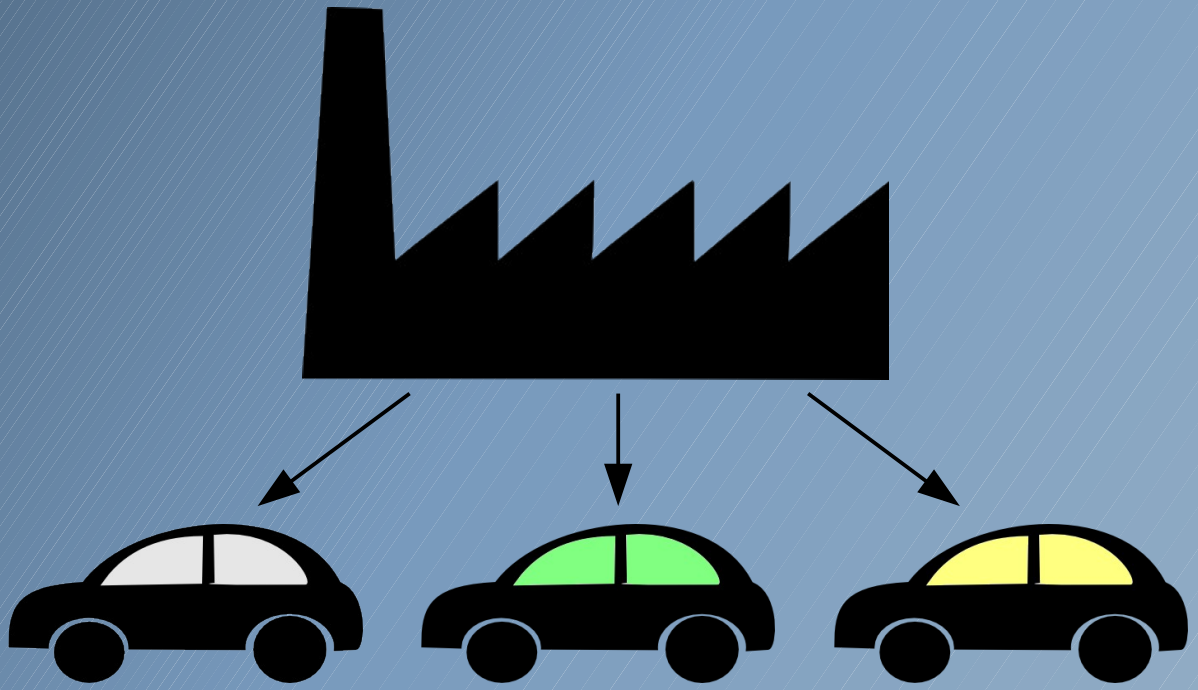


Envío de mensajes con clases



Clases como fábricas de objetos

- Proceso de instanciación.



```
miCaja := CajaDeAhorro new.
```

Formas de conocimiento

- Para que un objeto conozca a otro lo debe poder nombrar. Decimos que se establece una ligadura (*binding*) entre un nombre y un objeto.
- Podemos identificar 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.

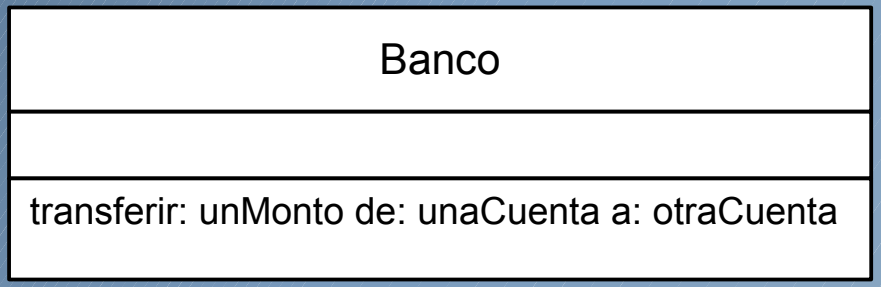
Variables de instancia

- Define una relación entre un objeto y sus colaboradores.
- Se definen explícitamente como parte de la estructura de la clase.
- La relación dura tanto tiempo como viva el objeto, aunque el objeto nombrado puede cambiar.
- ¿Pueden nombrar un ejemplo?

Parámetros

- Se refiere a los parámetros de un mensaje.
- El nombre de la relación se define explícitamente en el nombre del mensaje.
- La relación de conocimiento dura el tiempo que el método se encuentra activo.
- La ligadura entre el nombre y el objeto no puede alterarse durante la ejecución del método.

Parámetros - ejemplo



transferir: **unMonto** de: **unaCuenta** a: **otraCuenta**

unaCuenta extraer: **unMonto**.
otraCuenta depositar: **unMonto**.

Variables temporales

- Definen relaciones temporales dentro de un método.
- La relación con el objeto se crea durante la ejecución del método.
- El nombre de la relación se define explícitamente en el método.
- La relación dura el tiempo que el método se encuentra activo.
- Durante la ejecución del método, la ligadura entre el nombre y el objeto puede alterarse.

Variables temporales - Ejemplo

CajeroAutomatico
banco
depositar: unMonto en: unNumeroDeCuenta

```
depositar: unMonto en: unNumeroDeCuenta
        | cuenta |

        cuenta := banco obtenerCuenta: unNumeroDeCuenta.
        cuenta depositar: unMonto.
```

Variables globales

- Un objeto puede tener un nombre global.
- La convención es escribir los nombres con mayúsculas.
- El período de vida de estos nombres suele ser prolongado e ir mas allá de la vida de un objeto en particular.
- A medida que pasa el tiempo se puede modificar el binding del nombre con el objeto.

Variables globales - ejemplo

```
inicializar

        UnaGlobal := 10.
```

```
alCuadrado

        ^UnaGlobal * UnaGlobal.
```

```
sumarUno

        UnaGlobal := UnaGlobal + 1.
```

Pseudo-variable self

- ¿Cómo hace un objeto para mandarse un mensaje a sí mismo?
- *self* un nombre especial para que el objeto haga referencia a sí mismo.
- Este nombre no está definido como una relación de conocimiento, es implícito (decimos que es una pseudo-variable).
- En algunos lenguajes se lo denomina *this*, pero la semántica es exactamente la misma.

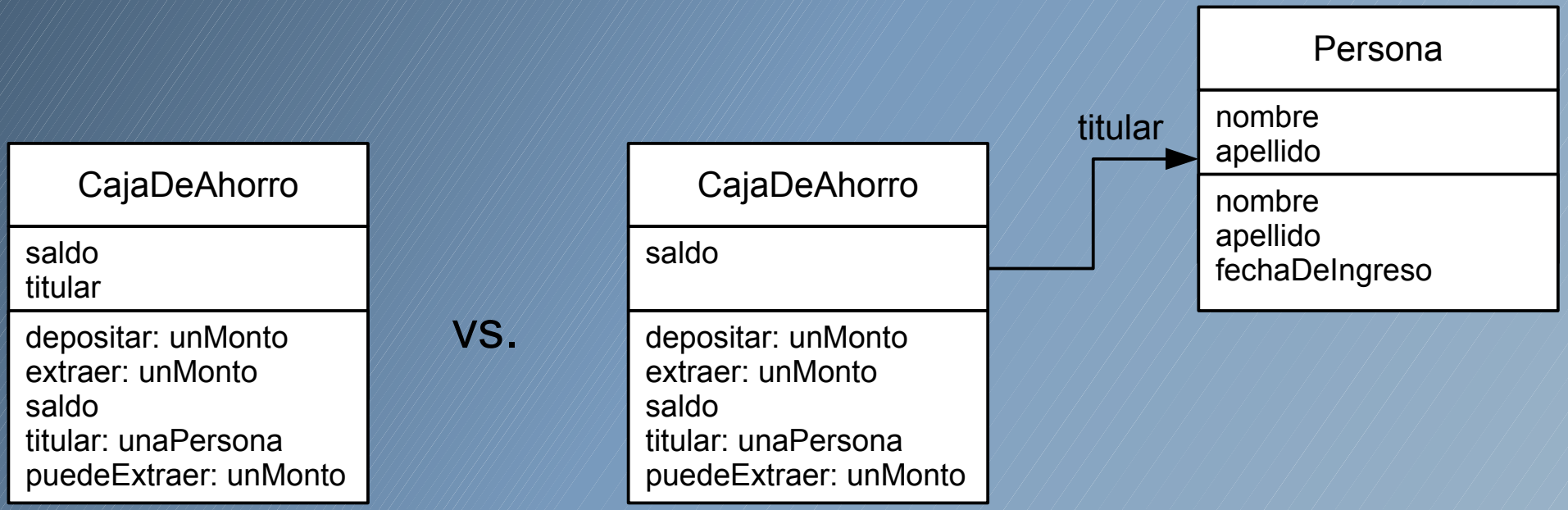
Formas de conocimiento - resumen

```
calcularCon: unNumero

    | temporal |

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

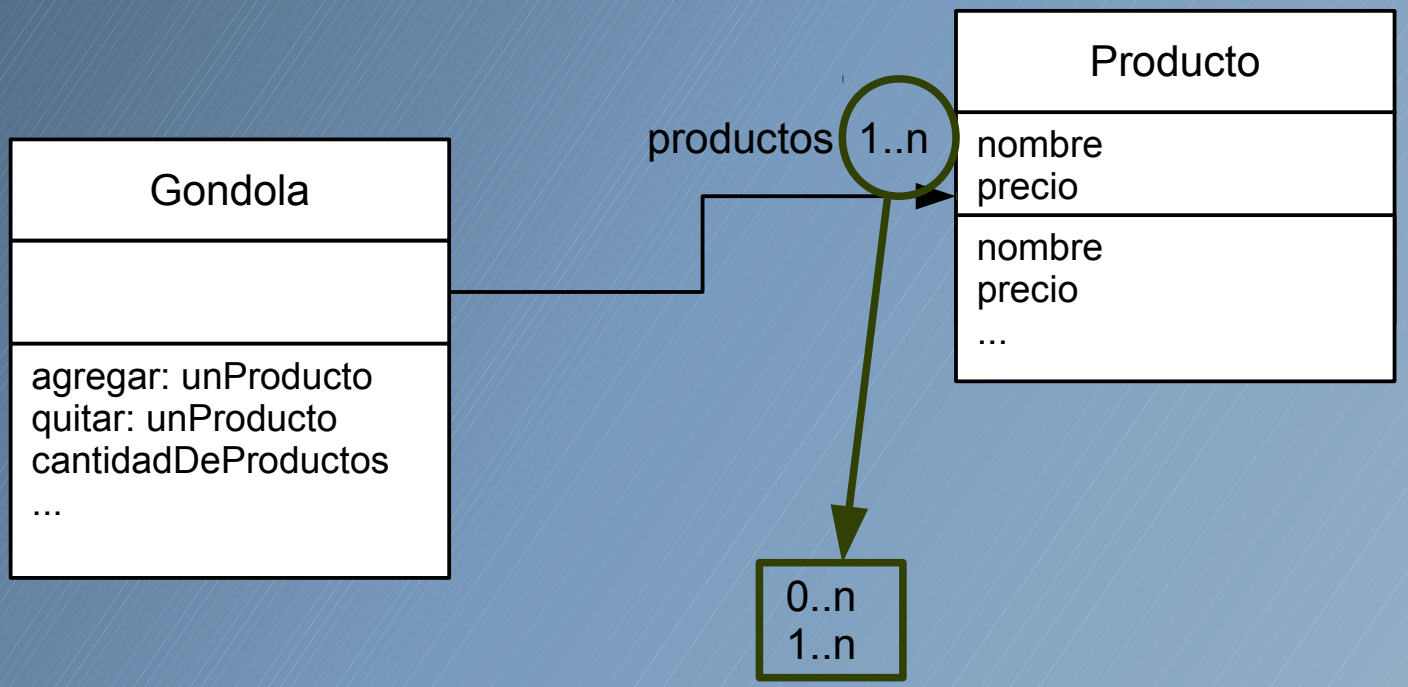
Calculador
velocidad
multiplicador calcularCon: unNumero

```
multiplicador

    ^ValorGlobal
```

Relaciones entre objetos



Relaciones entre objetos



Clase Fecha

- Ya modelamos la clase Fecha
  - `dia / dia: unNumero`
  - `mes / mes: unNumero`
  - `año / año: unNumero`
  - `= otraFecha`
  - Mensajes de comparación `(>, <, >=, <=)`.
  - `entre: unaFecha y: otraFecha`

Lapso de tiempo

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