Orientacion a Objetos 1

- **Profesores:**

  Dra. Alicia Diaz

  Dra. Roxana Giandini

  Dr. Gustavo Rossi

E-mails: [alicia, giandini, gustavo]@lifia.info.unlp.edu.ar

http://www.lifia.info.unlp.edu.ar/lifia/es/fichas/gustavo-rossi/

http://www.lifia.info.unlp.edu.ar/lifia/es/fichas/alicia-diaz/

http://www.lifia.info.unlp.edu.ar/lifia/es/fichas/roxana-giandini/

Practicas OO1 2016

- Inicio de prácticas: Semana del 29/8
- Explicación de Práctica:
  - **Lunes de 18 a 19hs - Aula 5**
  - **Miércoles de 11.15 a 12.15 hs - Aula 11**
  - **Elija una, se repiten los temas**
- Régimen de práctica
  - Asistencia no obligatoria.
  - Régimen de aprobación y promoción (se explicará en las exp. De práctica y se comunicará por moodle)
- Moodle como medio de comunicación y publicación de prácticas:
  - https://catedras.info.unlp.edu.ar/course/view.php?id=764
  - Suscripción automática (en base a SIU-Guaraní).

Contenidos del Curso

- Conceptos basicos de Objetos
- Introduccion a la Modelizacion
- Introduccion al desarrollo de sistemas de gran porte
- “Cultura” general respecto a lo que pasara en los proximos 5 años (al menos)

Conocimientos que suponemos: programación básica, concepto de variable, estructuras elementales, concepto de puntero

Contexto

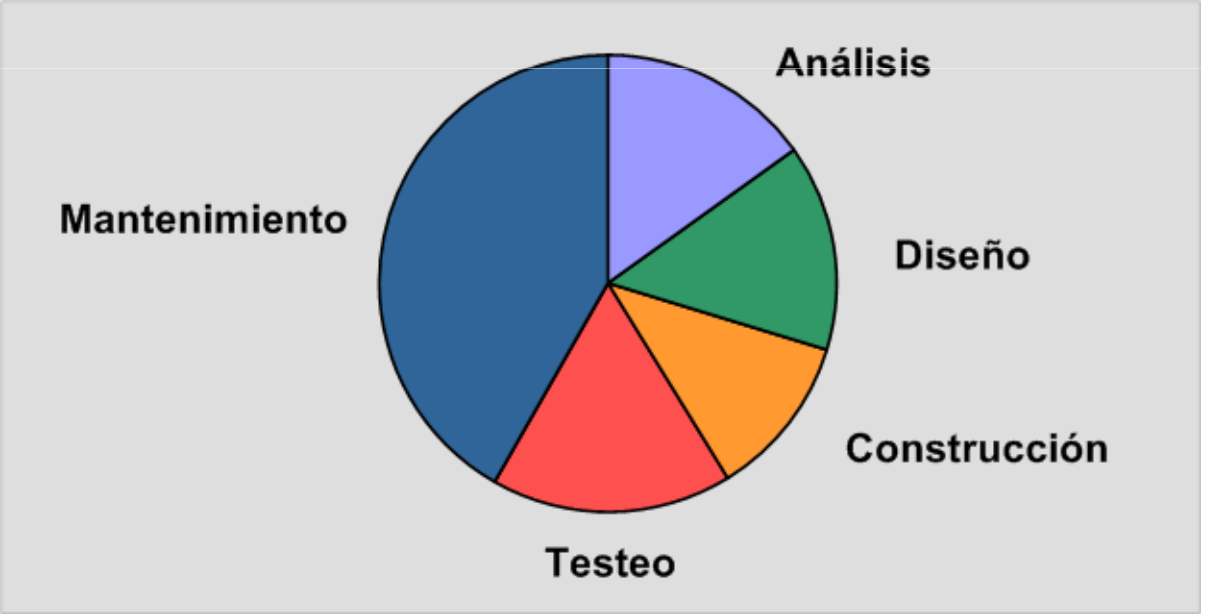
- En los 90: Computacion interactiva y Web Estatica
- En los 2000: Evolucion de la Web. Aparicion de la telefonia movil

En los 2010: Internet Movil, Internet de las cosas, Computacion en la Nube

En Software: Metodos Agiles vs Unificados, Desarrollo conducido por modelos, Software mas “volatil”, Requerimientos cambiantes permanentemente. Clientes y usuarios mejor formados y piden funcionalidad mas sofisticada

Motivación

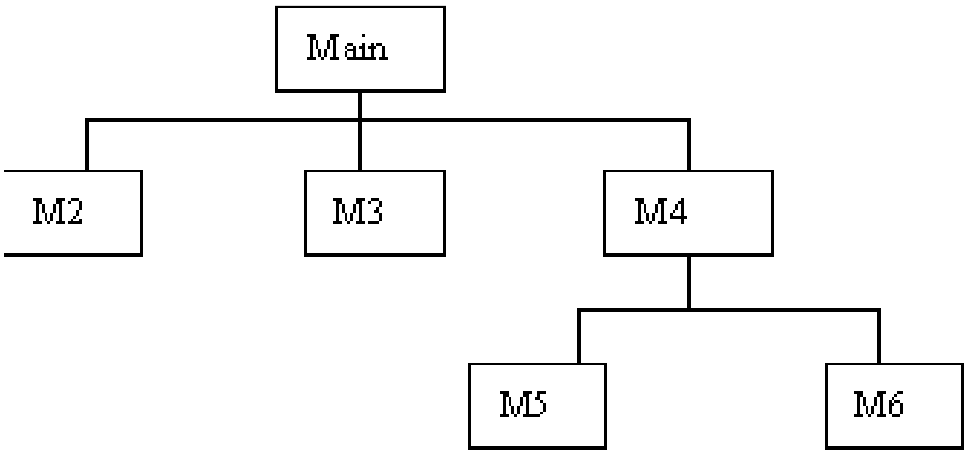
- Manejar complejidad
- Flexibilidad al cambio
- Mejorar la reusabilidad
- Minimizar costos de mantenimiento



Costo asociado a cada etapa del desarrollo

Programación Estructurada

- Sistemas contienen datos y programas.
- Los programas manipulan los datos.
- Los programas están organizados por:
  - Descomposición funcional.
  - Flujo de Datos.
  - Módulos.

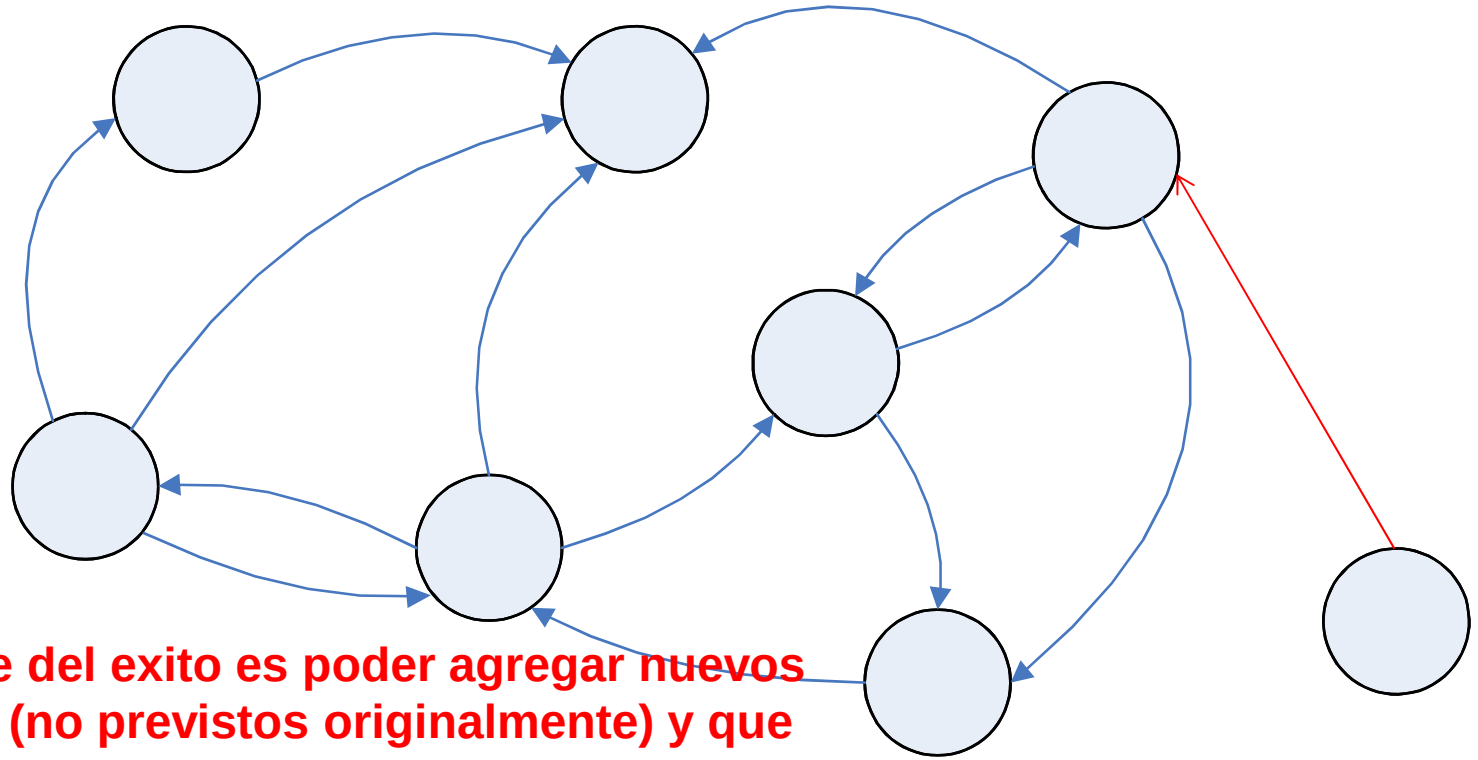


- Asignación, secuencia, iteración, condicionales

Programa Orientado a Objetos

- ¿Qué es un programa OO?

Un conjunto de ***objetos*** que ***colaboran*** enviándose ***mensajes***. Todo computo ocurre "dentro" de los objetos



La clave del exito es poder agregar nuevos objetos (no previstos originalmente) y que el sistema "no se entere"

Programación Orientada a Objetos

- Los sistemas están compuestos (solamente) por un conjunto de **objetos**.
- Los objetos son **responsables** de:
  - conocer sus propiedades,
  - conocer otros objetos (con los que colaboran) y
  - llevar a cabo ciertas acciones.
- Los objetos **colaboran** para llevar a cabo sus responsabilidades.

¿Qué es un objeto?

- Elemento primario que utilizamos para construir programas, desde lo más básico a lo más complejo.
- Es una ***abstracción*** de una ***entidad*** del ***dominio*** *del problema*. Ejemplos?
- *Puede representar conceptos del espacio de la solucion (estructuras de datos, tipos "basicos", archivos, ventanas, iconos..)*
- Un objeto tiene un ***comportamiento*** asociado.

Características de los Objetos

- Un objeto tiene:
  - ***Identidad.***
    - para distinguir un objeto de otro
  - ***Conocimiento.***
    - En base a sus relaciones con otros objetos y su estado interno
  - ***Comportamiento.***
    - Conjunto de mensajes que un objeto sabe responder

Identidad

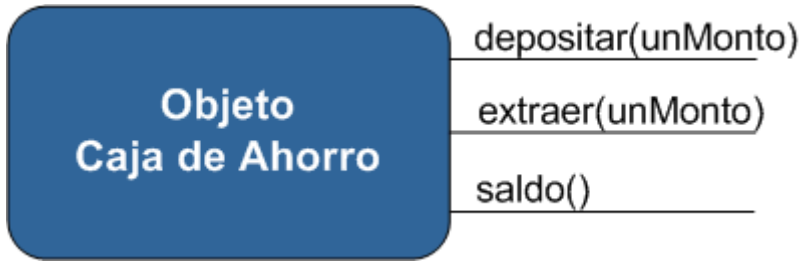
- Es una propiedad intrínseca de los objetos.
- Un objeto sólo es idéntico a sí mismo.
- No es lo mismo que *igualdad*.
  - Dos objetos pueden tener las mismas propiedades pero no son *el mismo* objeto

El estado interno

- El estado interno de un objeto determina su *conocimiento*.
- El estado interno está dado por:
  - Propiedades intrínsecas del objeto.
  - Relaciones con otros objetos con los cuales colabora para llevar a cabo sus responsabilidades.
- El estado interno se mantiene en las *variables de instancia (v.i.)* del objeto.
- Es **privado** del objeto. Ningún otro objeto puede accederlo.

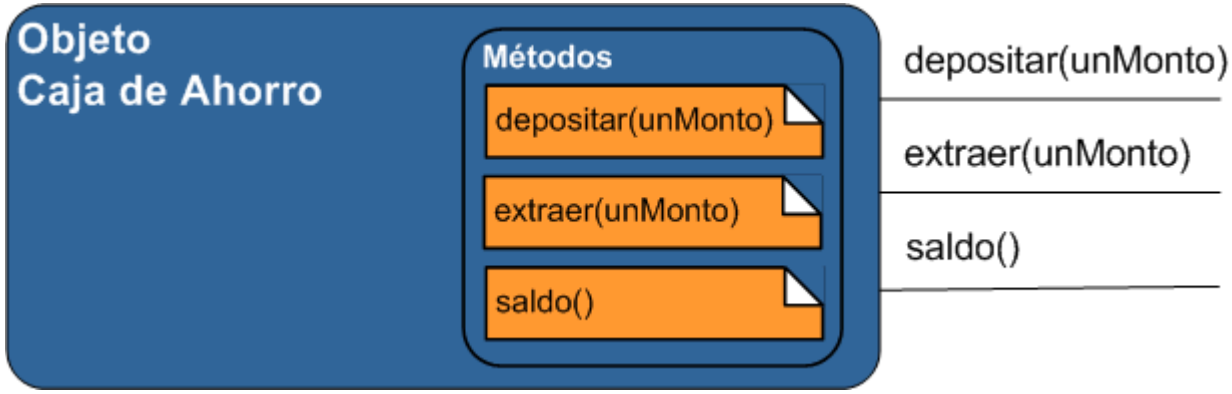
Comportamiento

- Un objeto se define en términos de su comportamiento.
- El comportamiento indica qué sabe hacer el objeto. Cuáles son sus ***responsabilidades***.
- Se especifica a través del conjunto de ***mensajes*** que el objeto sabe responder: ***protocolo***.
- Ejemplo:

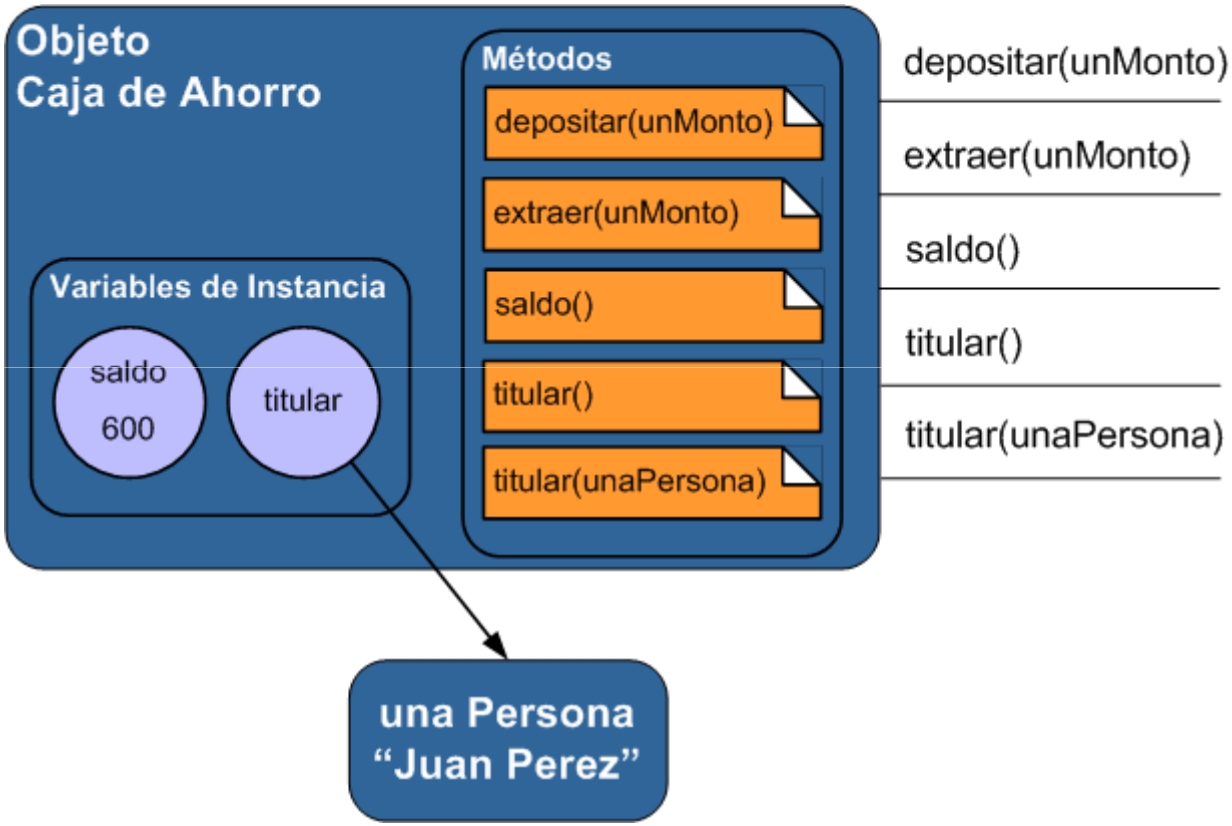


Comportamiento – implementación

- La **implementación** de cada mensaje (es decir, la manera en que un objeto responde a un mensaje) se especifica a través de un ***método***.
- Cuando un **objeto** recibe un **mensaje** responde activando el ***método*** asociado.
- El que envía el mensaje ***delega*** en el receptor la manera de resolverlo, que es ***privada*** del objeto.



Ejemplo Caja de Ahorro



Envío de un mensaje

- Para poder enviarle un mensaje a un objeto, hay que conocerlo.
- Al enviarle un mensaje a un objeto, éste responde activando el método asociado a ese mensaje (siempre y cuando exista).
- Como resultado del envío de un mensaje puede retornarse un objeto.

Especificación de un Mensaje

- ¿Cómo se especifica un mensaje?
  - ***Nombre***: correspondiente al protocolo del objeto receptor.
  - ***Parámetros***: información necesaria para resolver el mensaje.

- Cada lenguaje de programación propone una sintaxis particular para indicar el envío de un mensaje.
- A lo largo del curso utilizaremos la siguiente sintaxis:

```
<objeto receptor> <nombre de mensaje>: parámetros
```

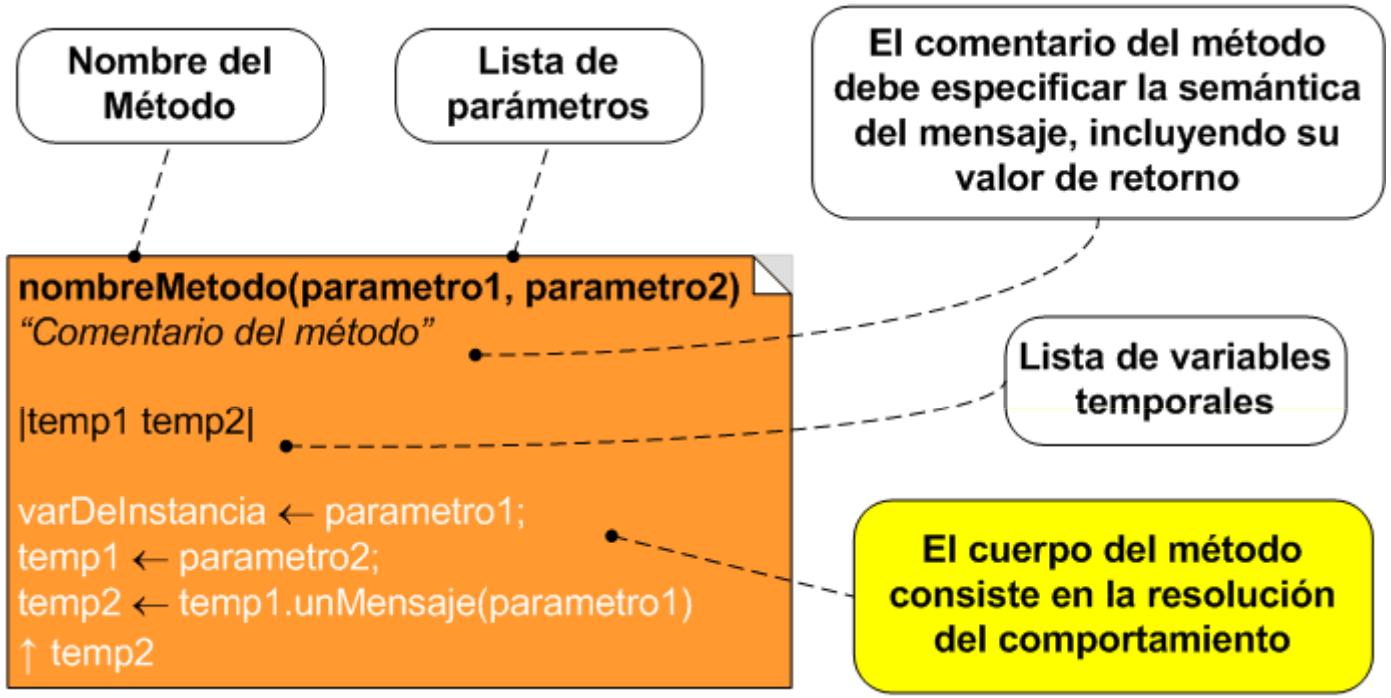
- Ejemplo empleando la sintaxis propuesta
  - Decirle a una cuenta bancaria que deposite $100 se escribe como:

```
unaCuenta depositar:100
```

Métodos

- ¿Qué es un método?
  - Es la contraparte funcional del mensaje.
  - Expresa la forma de llevar a cabo la semántica propia de un mensaje particular (el *cómo*).
- Un método puede realizar básicamente 3 cosas:
  - Modificar el estado interno del objeto.
  - Colaborar con otros objetos (enviándoles mensajes).
  - Retornar y terminar.

Especificación de un Método(pseudo lenguaje a la Java)



Ejemplo – Depositar en Cuenta Bancaria

depositar(unMonto)

"Agrega unMonto al saldo actual de la cuenta"

saldo ← saldo + unMonto

Ejemplo - Cajero

realizarDeposito(unMonto, nroCuenta)

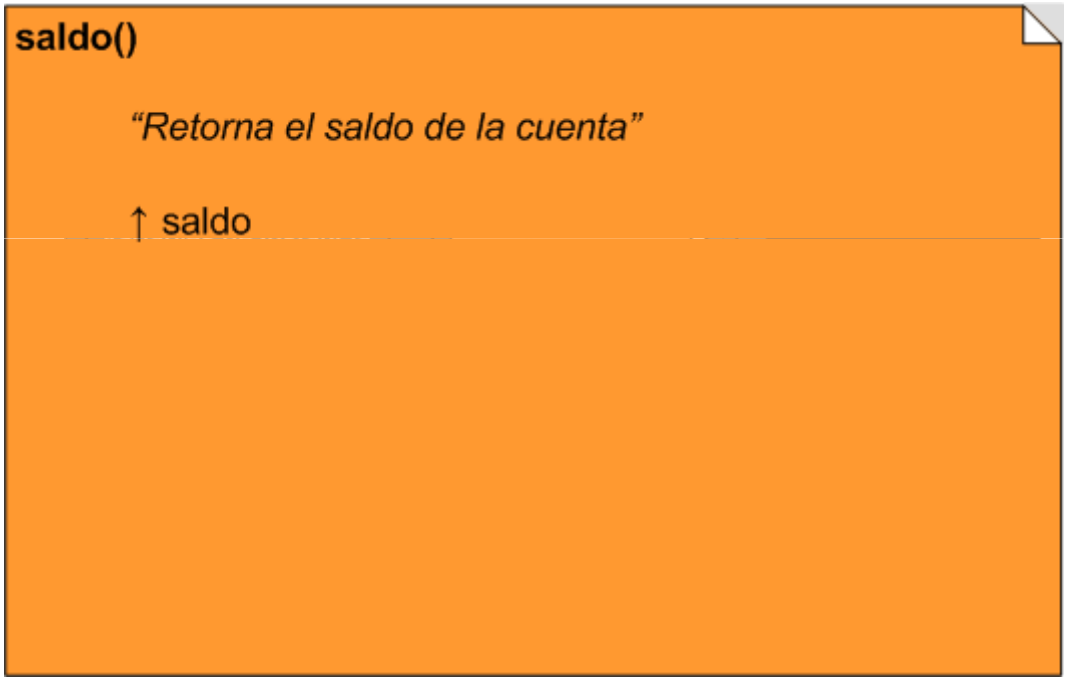
"Deposita unMonto en la cuenta numero nroCuenta"

|cuenta|

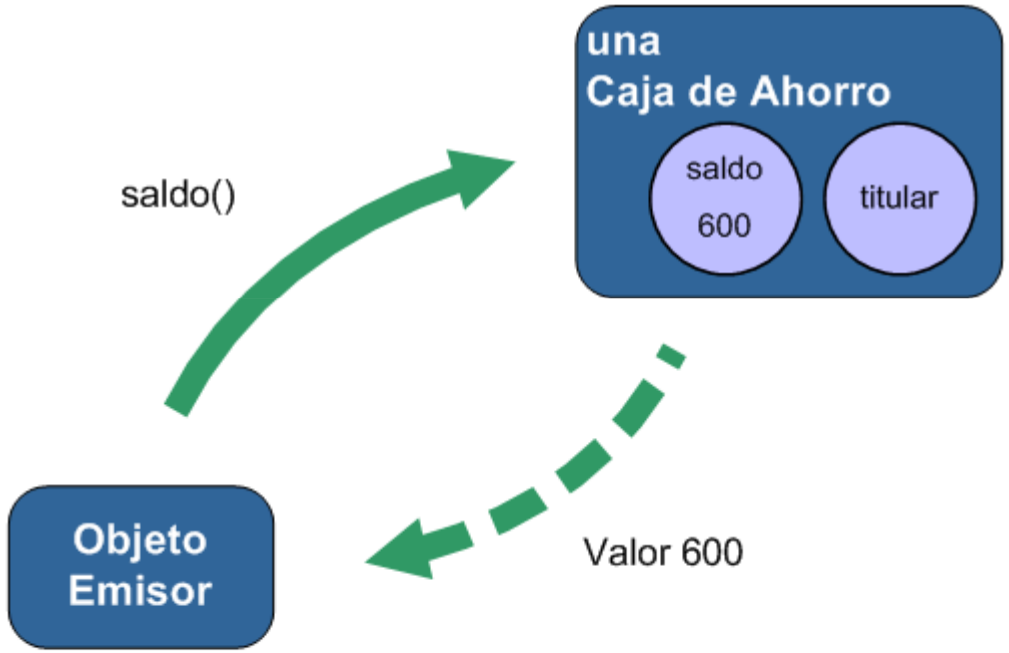
cuenta ← banco.buscarCuenta(nroCuenta);
cuenta.despositar(unMonto)

Saldo

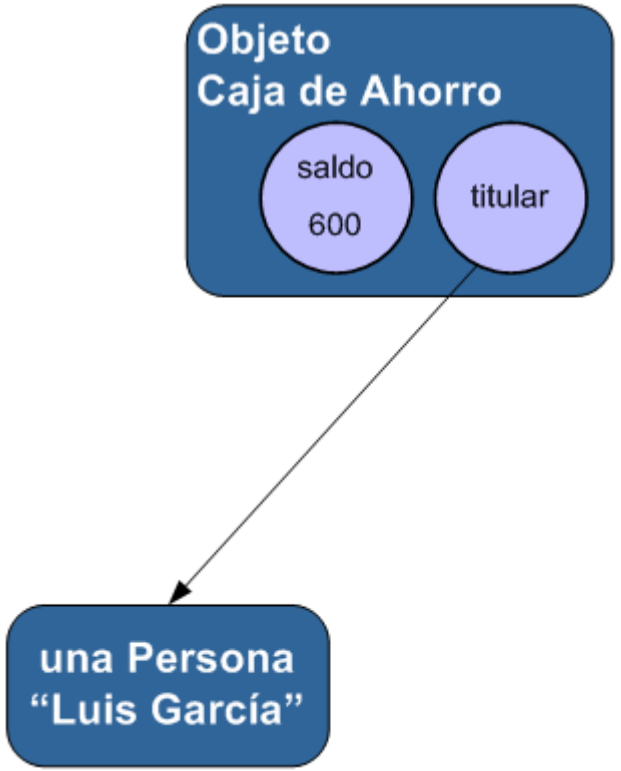
- ¿Cómo sería el método del mensaje **`saldo`**?



Envío del mensaje saldo

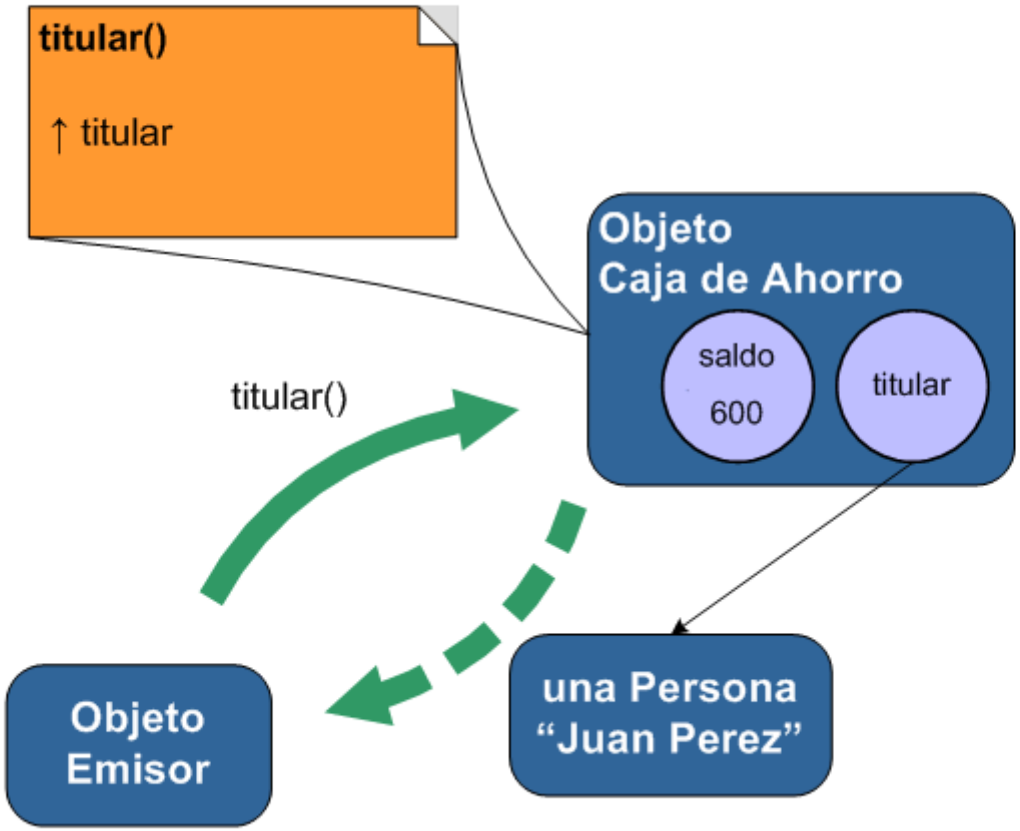


Cambio del estado interno



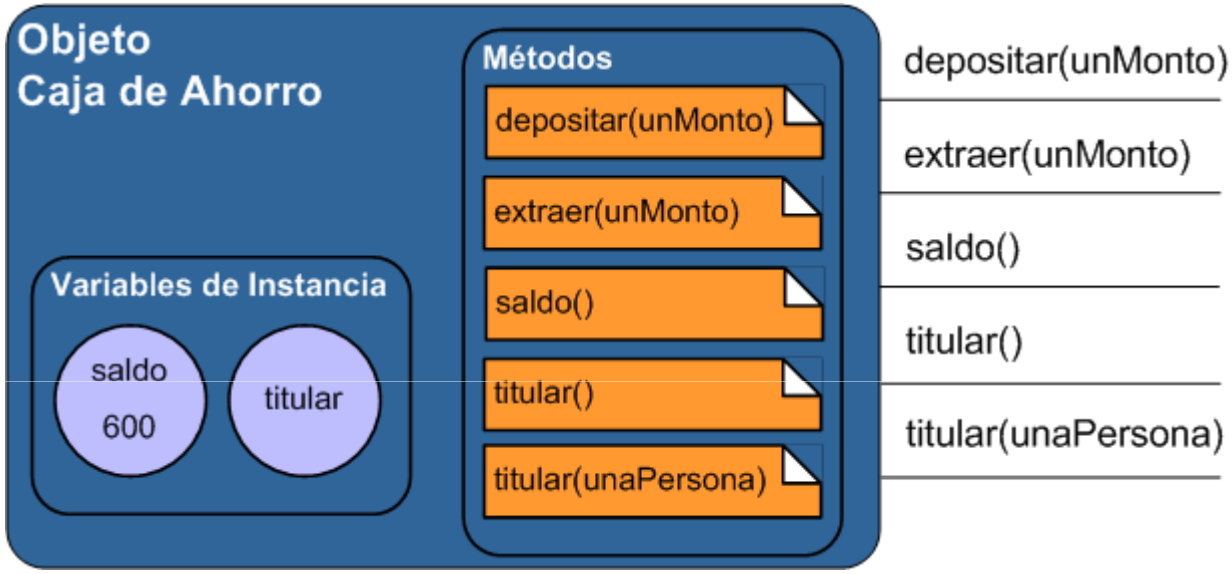
Ejemplo - Retornar el titular

- ¿Cómo sería el método del mensaje **`titular()`**?



Ejercicio – Depósito y Extracción

- ¿Cómo escribimos el método **`extraer(unMonto)`**?



depositar(unMonto)

"Agrega unMonto al saldo actual de la cuenta"

saldo ← saldo + unMonto

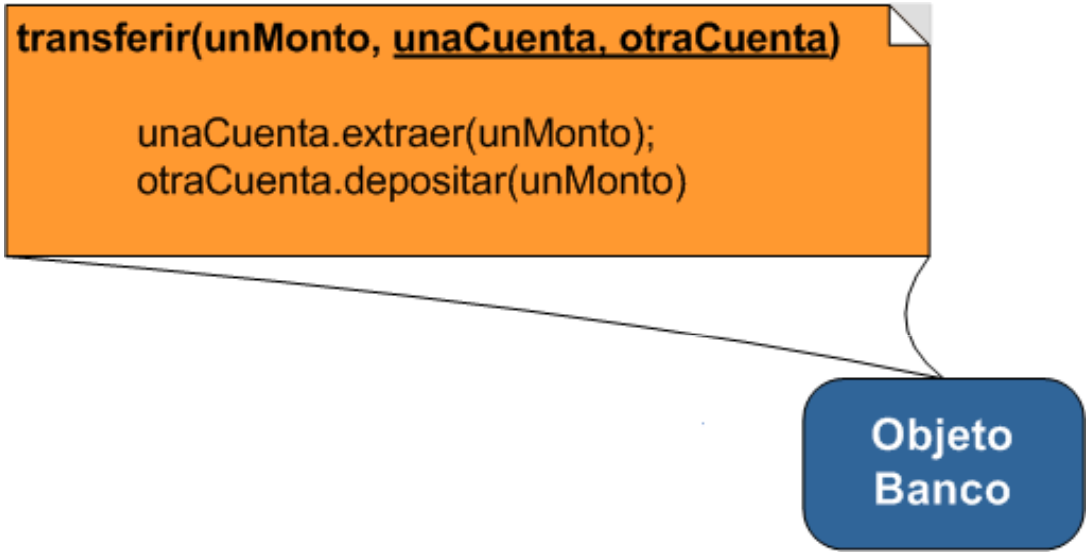
Formas de Conocimiento

- 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 tres formas de conocimiento o tipos de relaciones entre objetos.
  - Conocimiento Interno: Variables de instancia.
  - Conocimiento Externo: Parámetros.
  - Conocimiento Temporal: Variables temporales.
- Además existe una cuarta forma de conocimiento especial: las pseudo-variables.

Parámetros

- Se refiere a los parámetros de un 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.

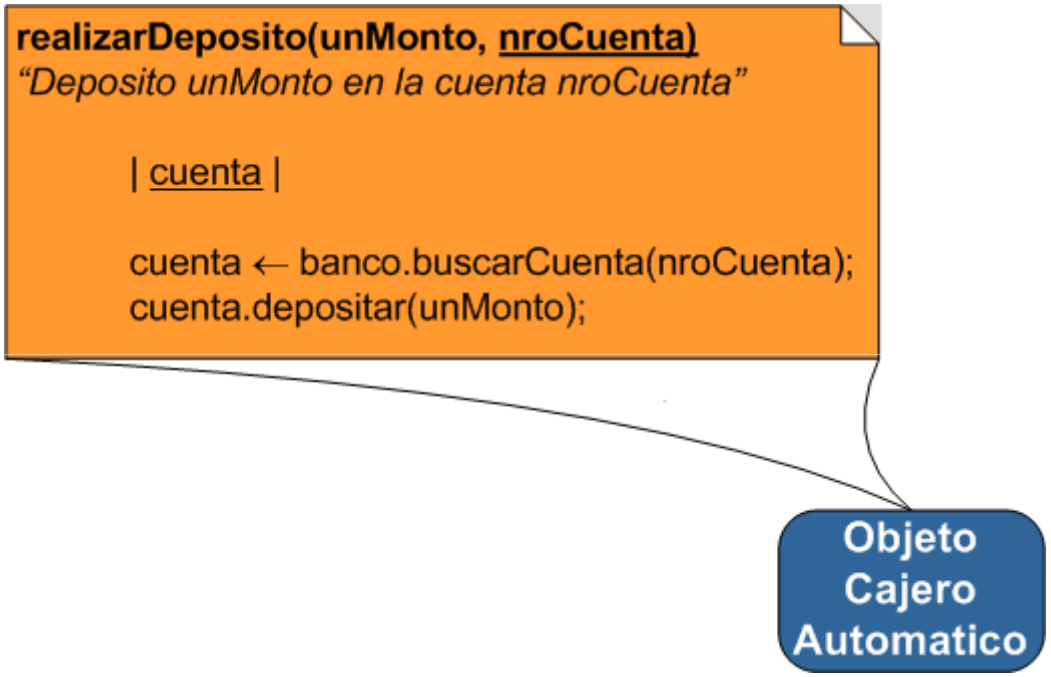
Parámetros


transferir(unMonto, unaCuenta, otraCuenta)
unaCuenta.extraer(unMonto);
otraCuenta.depositar(unMonto)
Objeto
Banco

Variables temporales

- Definen relaciones temporales dentro de un método.
- La relación con el objeto se crea durante la ejecución del método.
- La relación se mantiene dentro del contexto donde fue definida la variable.
- Durante la ejecución del método, la ligadura entre el nombre y el objeto puede alterarse.

Variables temporales



Encapsulamiento

"Es la cualidad de los objetos de ocultar los detalles de implementación y su estado interno del mundo exterior"

- Características:
  - Esconde detalles de implementación.
  - Protege el estado interno de los objetos.
  - Un objeto sólo muestra su "cara visible" por medio de su protocolo.
  - Los métodos y su estado quedan escondidos para cualquier otro objeto. Es el objeto quien decide *qué* se publica.
  - Facilita modularidad y reutilización.

Encapsulamiento