

Contexto

- En los 90: Aplicaciones interactivas y Web 1.0
- En los 2000: Evolución de la Web. Aparición de la telefonía móvil
- A partir del 2010: Internet Móvil, Internet de las cosas, Computación en la Nube, Big Data, I.A.

En Software: Métodos Agiles vs Monolíticos, Desarrollo conducido por modelos, Software mas “volátil”, Requerimientos cambiantes permanentemente.

¿Qué tipo de aplicaciones construimos hoy?

- Distribuidas (combinando hard/soft, personas...)
- Basadas en “servicios” provistos por terceros
- Que combinan “partes” de otras aplicaciones
- Que pueden crecer en forma completamente inimaginable..
- Con operaciones que no son atómicas (ej. Invitación de amistad en Facebook)
- Que se componen y descomponen y cuyas partes son usadas por otros....
- Negocios completamente basados en el software

Como las construimos? Como las construimos?

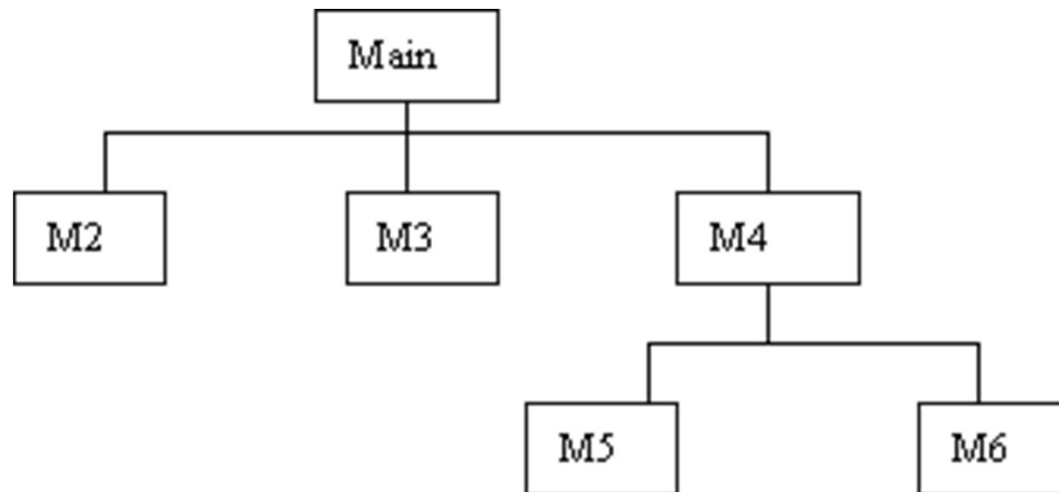
- Antes: Análisis, Diseño, Programación, Testing (Cascada)
- Hoy: Un poco de análisis, Un poco de diseño, Un poco de programación y testing (y empezamos de nuevo) (Agile)
- Pero....y la Arquitectura?

Aplicaciones complejas...

- ¿Cómo descomponemos?
- ¿Cómo decidimos que componentes usar y como distribuirlos?
- ¿Cómo los estructuramos una vez tomada esa decisión?
- ¿En qué bloques de construcción nos basamos?
- ¿Cómo salimos de la “tiranía” del main monolítico?
- ¿Cómo nos volvemos “poliglotas” sin volvernos locos? (¿cuántos lenguajes diferentes usa una aplicación compleja?)

Programación Estructurada

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

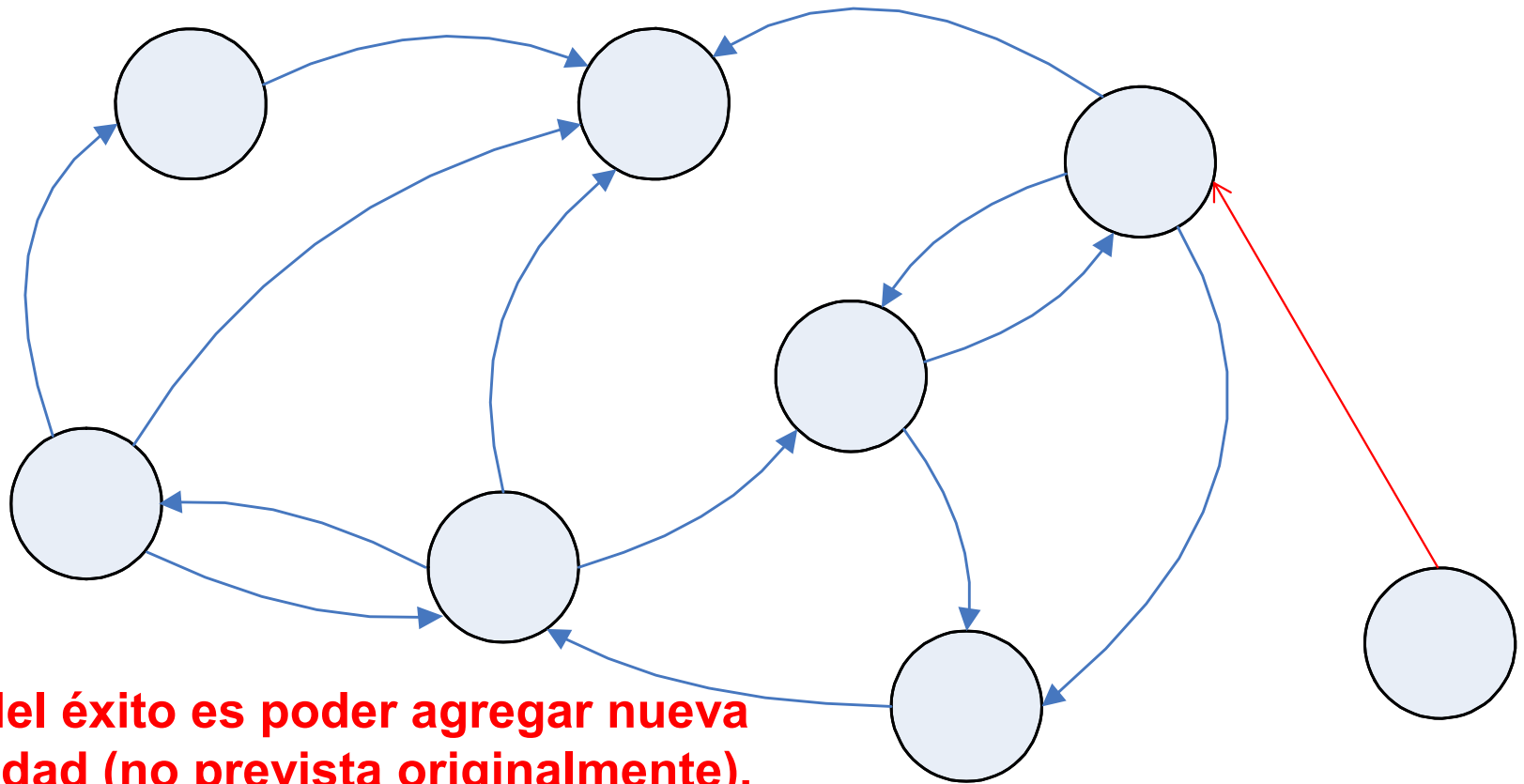


- Asignación, secuencia, iteración, condicionales

Programa o Sistema Orientado a Objetos

- ¿Como es un software construido con objetos?

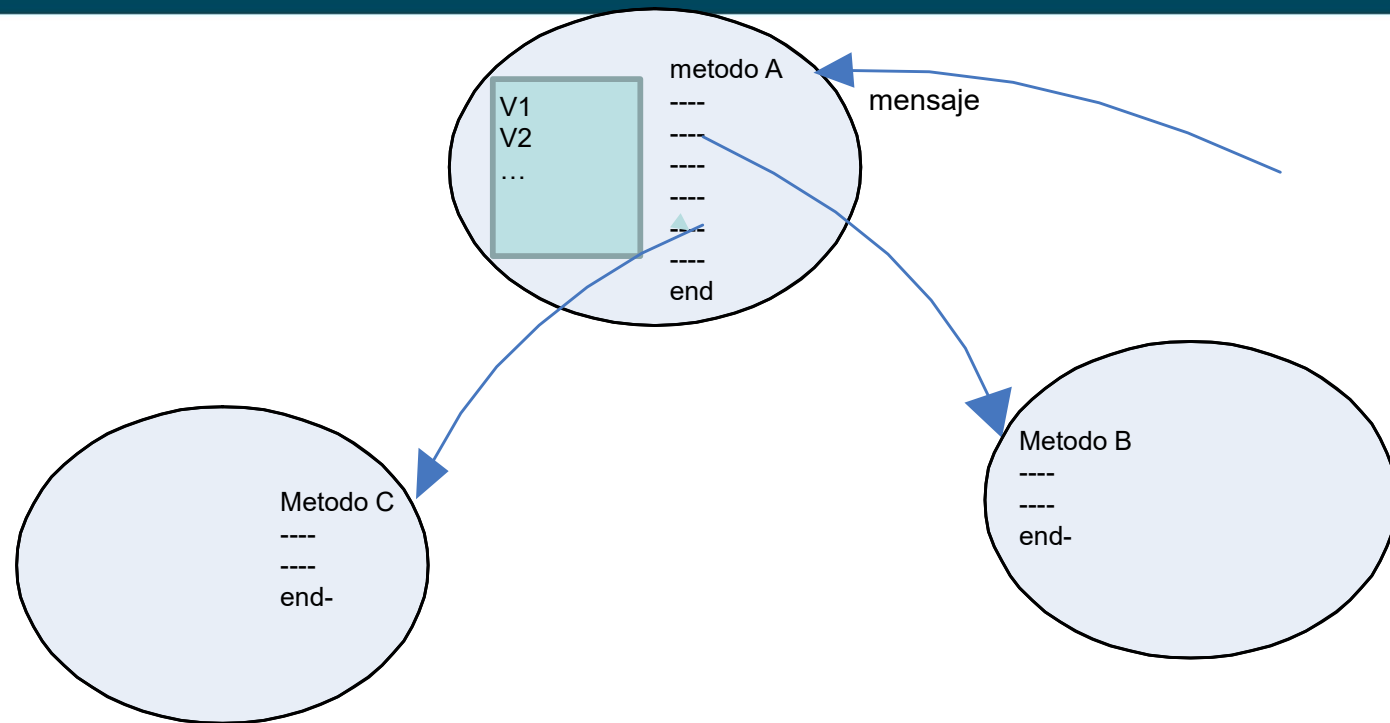
Un conjunto de *objetos* que *colaboran* enviándose *mensajes*. Todo computo ocurre “dentro” de los objetos



La clave del éxito es poder agregar nueva funcionalidad (no prevista originalmente), reemplazar objetos o modificar objetos y que el sistema “no se entere”, ni se rompa.

E.g. integración Whatsapp y messenger Facebook

Más en detalle



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

Aspectos de interés en esta definición

- No hay un objeto “main”
- Cuando codificamos, describimos (programamos) clases
- Una jerarquía de clases no indica lo mismo que la jerarquía top-down
- Cuando se ejecuta el programa lo que tenemos son objetos que cooperan y que se crean dinámicamente durante la ejecución del programa

Aspectos de interés....

- Podemos pensar la interacción usuario/software de la misma manera
- Este mismo modelo nos permite entender (al menos en parte) otros modelos de computación: viendo a los objetos como proveedores de servicios por ejemplo
- Este mismo modelo no asume objetos localizados en el mismo espacio de memoria (pueden estar distribuidos)

Impacto en como “pensamos” el software

- La estructura general cambia: en vez de una jerarquía: Main/procedures/sub-procedures tenemos una red de “cosas” que se comunican
- Pensamos en que “cosas” hay en nuestro software (los objetos) y como se comunican entre sí.
- Hay un “shift” mental crítico en forma en la cual pensamos el software como objetos
- Mientras que la estructura sintáctica es “lineal” el programa en ejecución no lo es

¿Qué es un objeto?

- Es una *abstracción* de una *entidad* del *dominio del problema*. Ejemplos: Persona, Producto, Cuenta Bancaria, Auto, Plan de Estudios,....
- *Puede representar también conceptos del espacio de la solución (estructuras de datos, tipos “básicos”, archivos, ventanas, iconos..)*

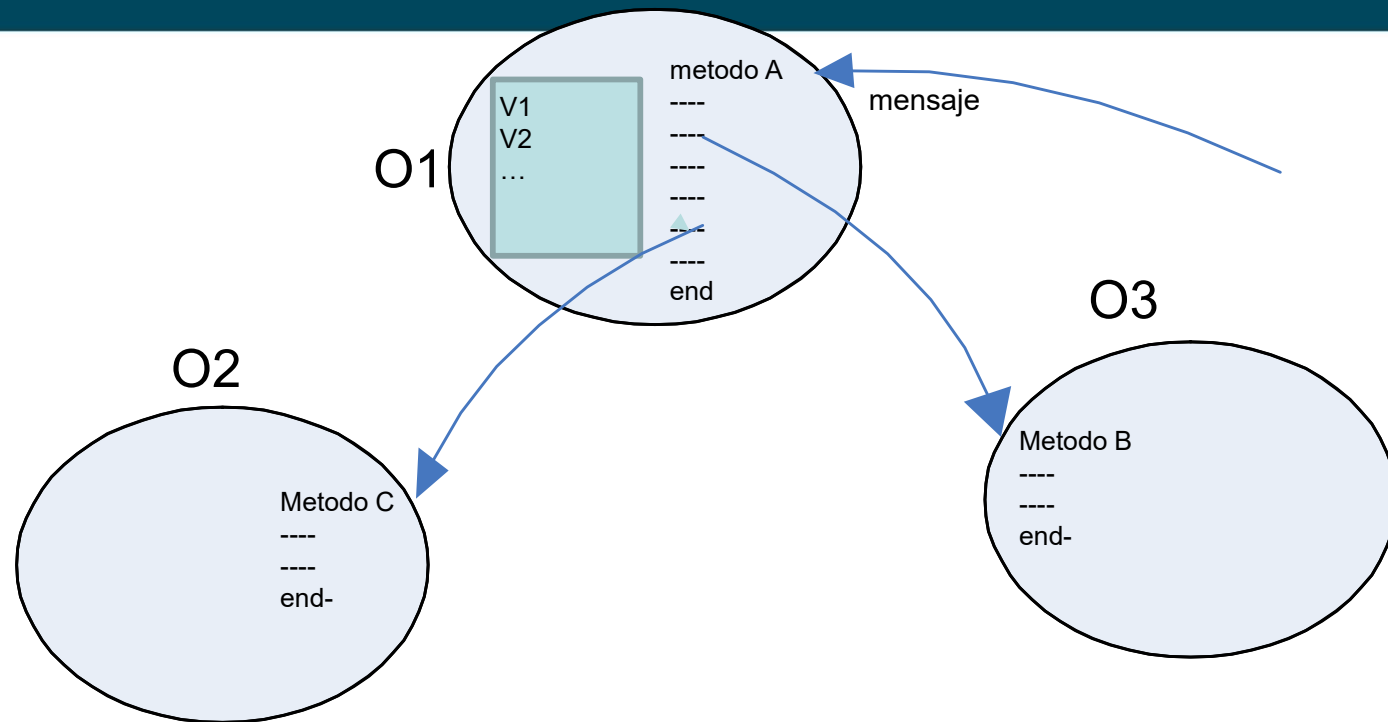
Características de los Objetos

- 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

El estado interno

- El estado interno de un objeto determina su *conocimiento*.
- El estado interno esta dado por:
 - Propiedades básicas (intrínsecas) del objeto.
 - 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. (¿Cuál es el impacto de esto?)

Variables de instancia



- En general las variables son REFERENCIAS (punteros) a otros objetos con los cuales el objeto colabora.
- Algunas pueden ser atributos básicos
- En el grafico O1 puede mandarle mensajes a O2 y O3 porque “los conoce”, o sea hay una variable en O1 que APUNTA a O2 y otra a O3 (o la misma variable que cambia de valor en diferentes momentos)

Ejemplos

- Objeto Alumno:

v.i: nombre, dni, fecha nac,
carrera, legajo

-Las 2 primeras pueden ser tipos “básicos”: string, numero (dependiendo del lenguaje)

-Las otras 3 son referencias a objetos de la Clase Fecha, Carrera y Legajo

- Objeto Carrera:

v.i: nombre, año, materias

Materias es una colección (¿array?) de objetos Materia

- Objeto Materia:

v.i: nombre, año, semestre, correlativas

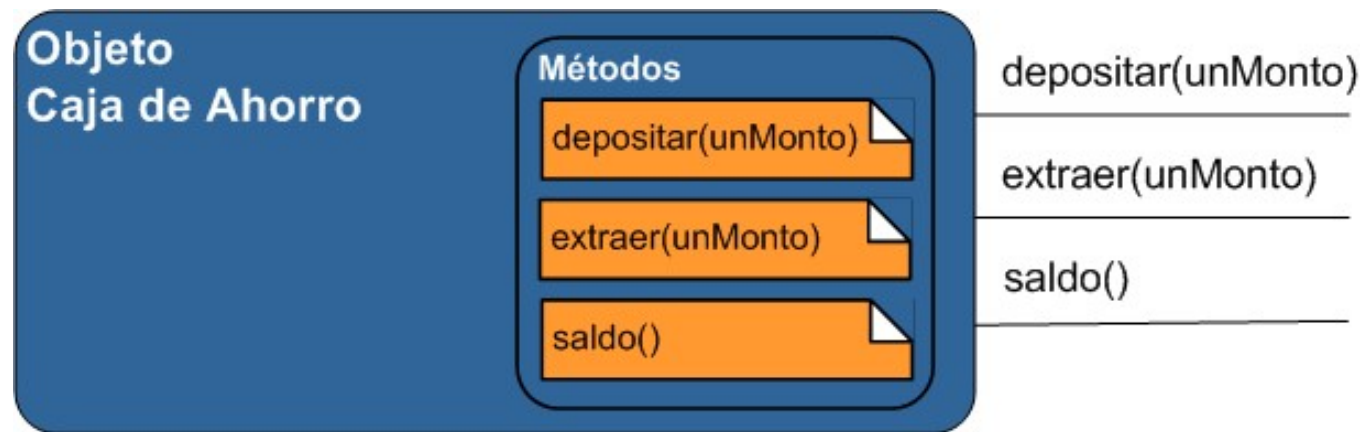
Comportamiento

- 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:



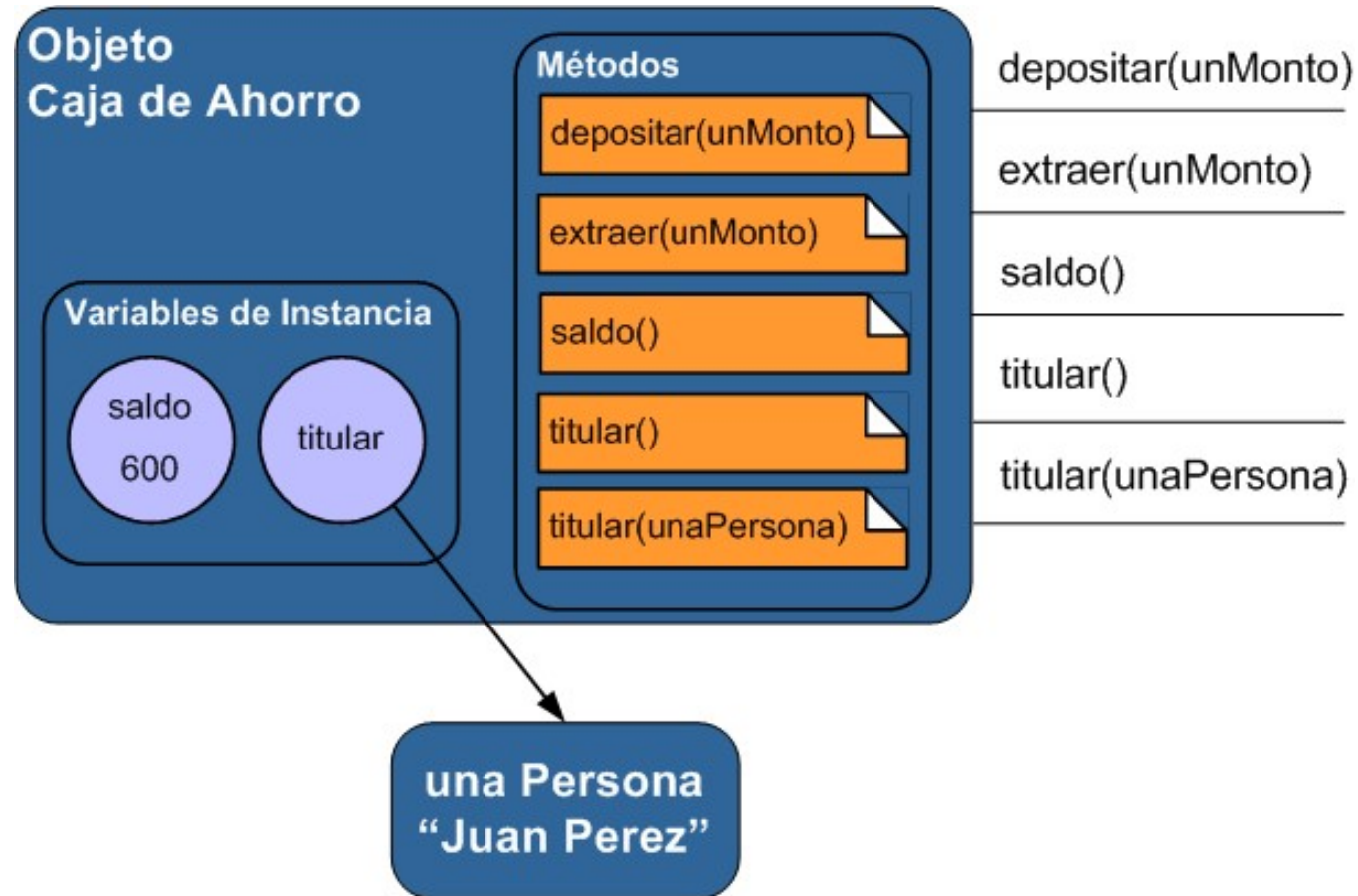
Comportamiento - implementación

- La **realización** 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.



Impacto: Localizacion de funcionalidad

Ejemplo Caja de Ahorro



Obsérvese la variable “titular” apuntando a un objeto Persona

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.

Ejemplo: cuenta.depositar(cantidad)

figura.dibujar()

figuraGrande.rotar(45)

- *cuenta*, *figura*, *figuraGrande* son variables que apuntan a un objeto que entiende el mensaje correspondiente

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.

Y la entrada/salida de informacion?

- En un sistema diseñado correctamente, un objeto (programado por el desarrollador) no debería realizar ninguna operación vinculada a la interfaz (mostrar algo) o a la interacción (esperar un “input”)
- En la mayoría de los entornos de desarrollo es hasta imposible hacerlo, y en nuestro caso lo será
- ¿Qué ganamos? Poder cambiar el estilo o el dispositivo de interacción sin necesidad de tocar el Código que pasa a ser independiente de la interfaz

Ejemplo en Java - Métodos en Cuenta Bancaria

```
public double getSaldo() {  
    return saldo;  
}
```

```
public void depositar (double monto) {  
    saldo = saldo + monto;  
}
```

```
public void extraer(double monto) {  
    saldo = saldo - monto;  
}
```

```
public void transferir(double monto, CajaDeAhorro cuentaDestino) {  
    saldo = saldo - monto;  
    cuentaDestino.depositar(monto);  
}
```

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 (como “this” o “self”)

“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.

Clases e instancias

- 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*
- En consecuencia todas las instancias de una clase se comportan de la misma manera.
- Cada instancia mantendrá su propio estado interno.

Especificación de Clases

- Las clases se especifican por medio de un nombre, el estado o estructura interna que tendrán sus instancias y los métodos asociados que definen el comportamiento
- Gráficamente:

Variables de Instancia

Los nombre de las v.i. se escriben en minúsculas y sin espacios

CajaDeAhorro
saldo titular
depositar(unMonto) extraer(unMonto) saldo() titular() titular(unaPersona) sePuedeExtraer(unMonto)

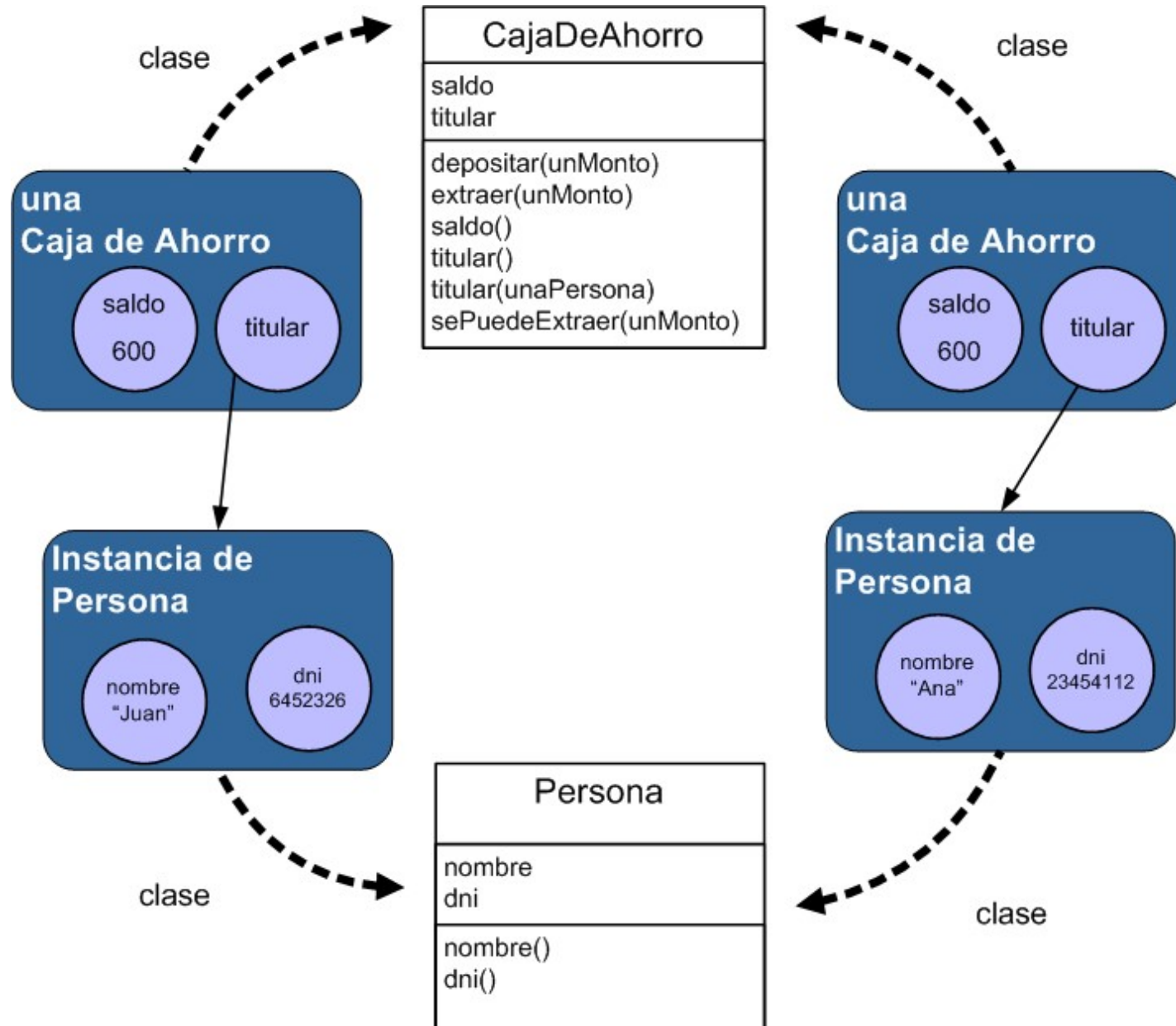
Nombre de la Clase

Comienzan con mayúscula y no posee espacios

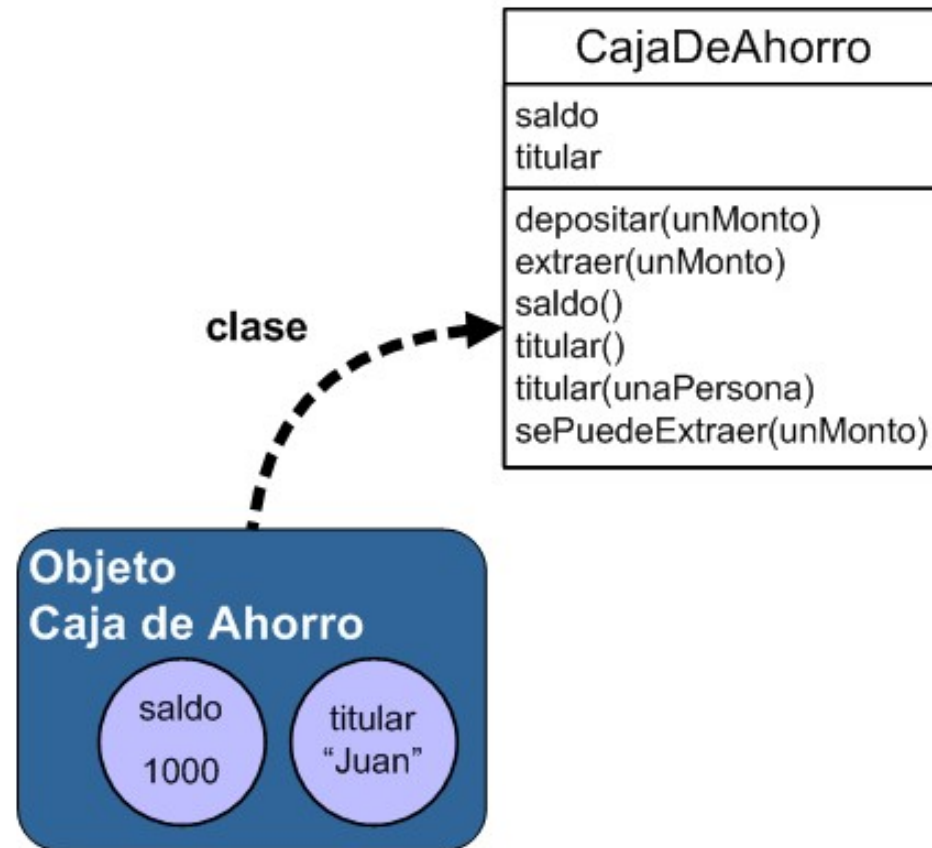
Protocolo

Para cada mensaje se debe especificar como mínimo el nombre y los parámetros que recibe

Ejemplo de clases e instancias

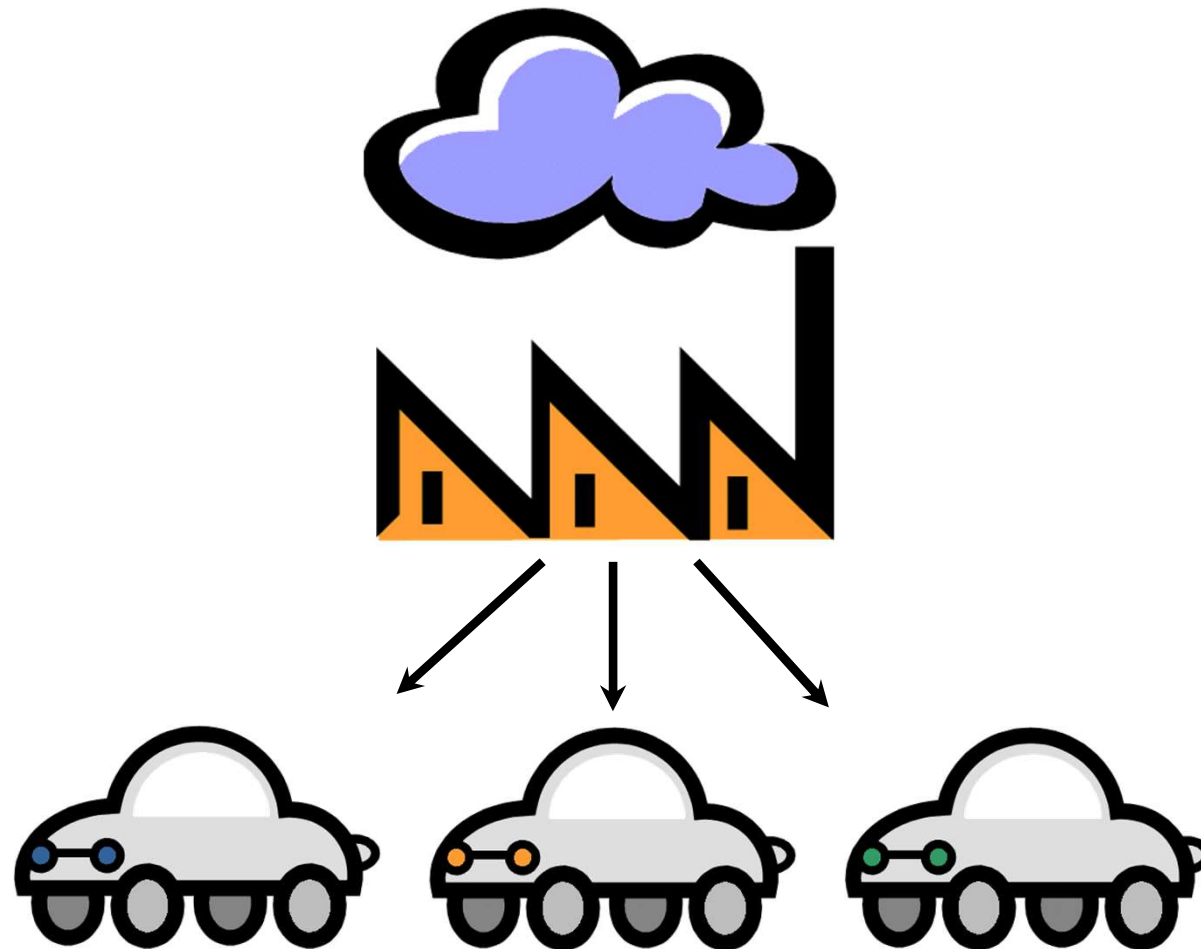


Envío de mensajes con clases



Creación de Objetos

- ¿Cómo creamos nuevos objetos?
- Instanciación



Creación de Objetos

- Comúnmente se utiliza la palabra reservada **new** para instanciar nuevos objetos.
- ¿Quién crea objetos? ¿Cuándo los crea?

```
/** Método en alguna otra clase del sistema */  
public CajaDeAhorro abrirCajaDeAhorro(Cliente titular  
    , double depositoInicial)  
{  
    CajaDeAhorro nuevaCuenta = new CajaDeAhorro(titular);  
    nuevaCuenta.depositar(depositoInicial);  
    cuentas.add(nuevaCuenta);  
    return nuevaCuenta;  
}
```

Instanciación

- Es el mecanismo de creación de objetos.
- Los objetos se *instancian* a partir de un molde.
- La **clase** funciona como molde.
- Un nuevo objeto es una *instancia* de una clase.
- Todas las instancias de una misma clase
 - Tendrán la misma estructura interna.
 - Responderán al mismo protocolo (los mismos mensajes) de la misma manera (los mismos métodos).

- Para que un objeto esté listo para llevar a cabo sus responsabilidades hace falta *inicializarlo*
- Inicializar un objeto significa darle valor a sus variables
- ¿De dónde sacamos esos valores iniciales?

Constructores

```
public CajaDeAhorro(Cliente unCliente) {  
    saldo = 0;  
    titular = unCliente;  
}
```

```
public CajaDeAhorro(Cliente unCliente,  
                    double saldoInicial) {  
    saldo = saldoInicial;  
    titular = unCliente;  
}
```