Orientacion a Objetos 1

- **Profesores:**

Dra. Roxana Giandini

Dr. Alejandro Fernandez

Dr. Gustavo Rossi

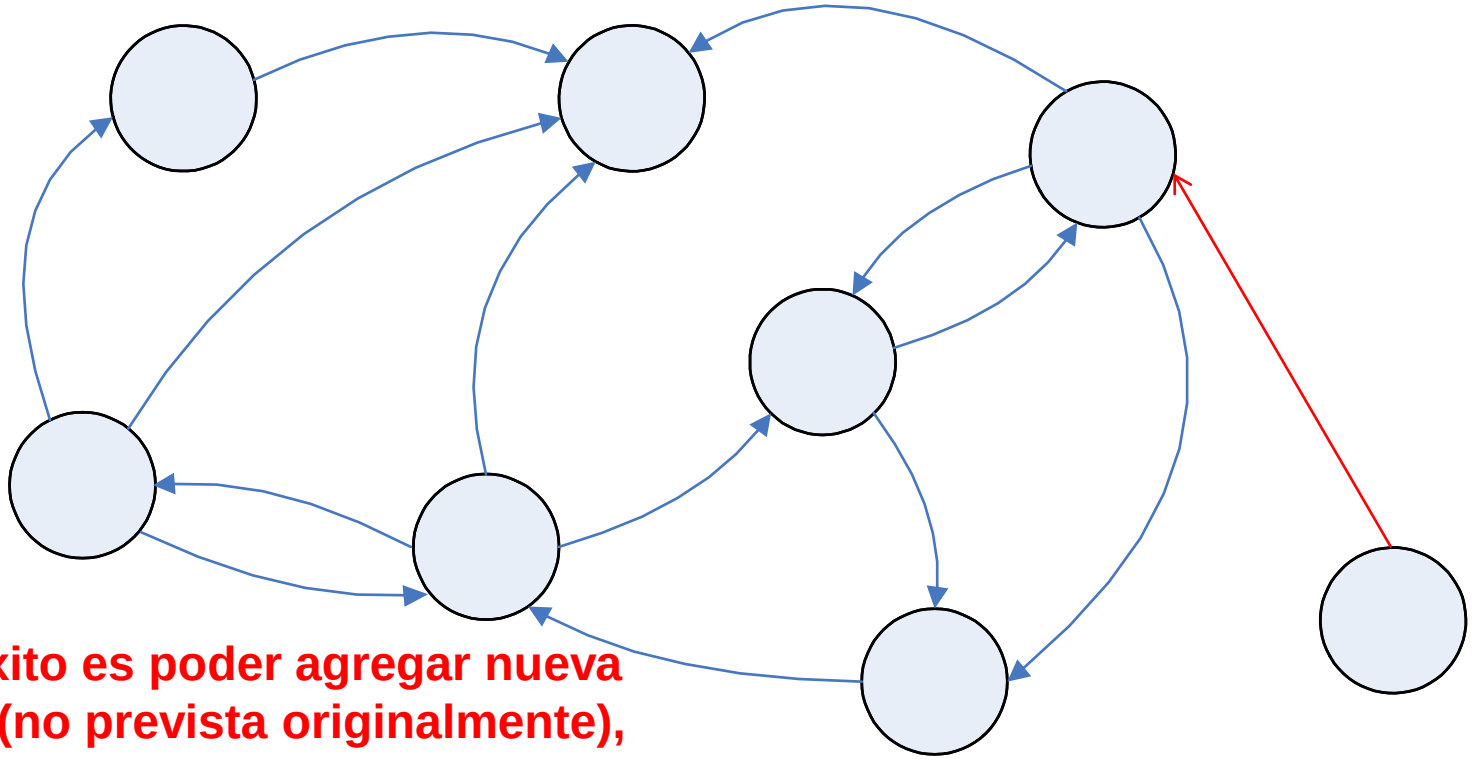
email:

[giandini, gustavo, alejandro.fernandez]@lifia.info.unlp.edu.ar

Programa o Sistema Orientado a Objetos

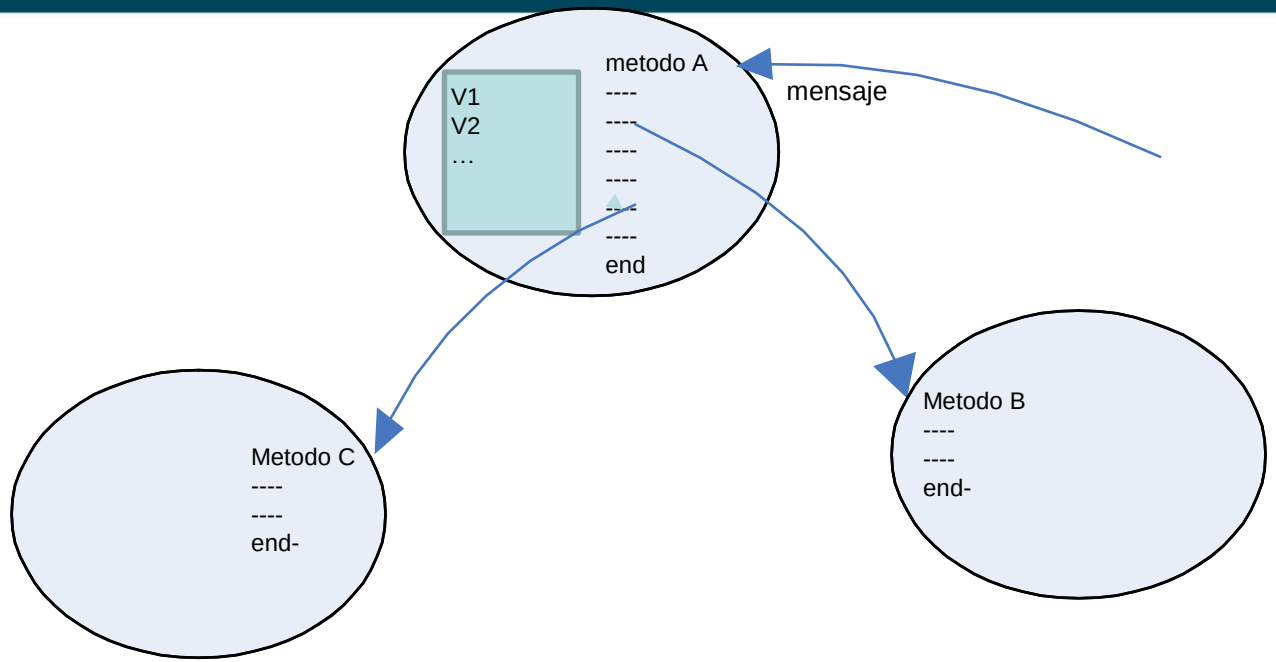
- ¿Como es un software construido con objetos?

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



La clave del exito 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. integracion 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 jerarquia de clases no indica lo mismo que la jerarquia 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 ejecucion no lo és

¿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 tambien conceptos del espacio de la solucion (estructuras de datos, tipos "basicos", 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

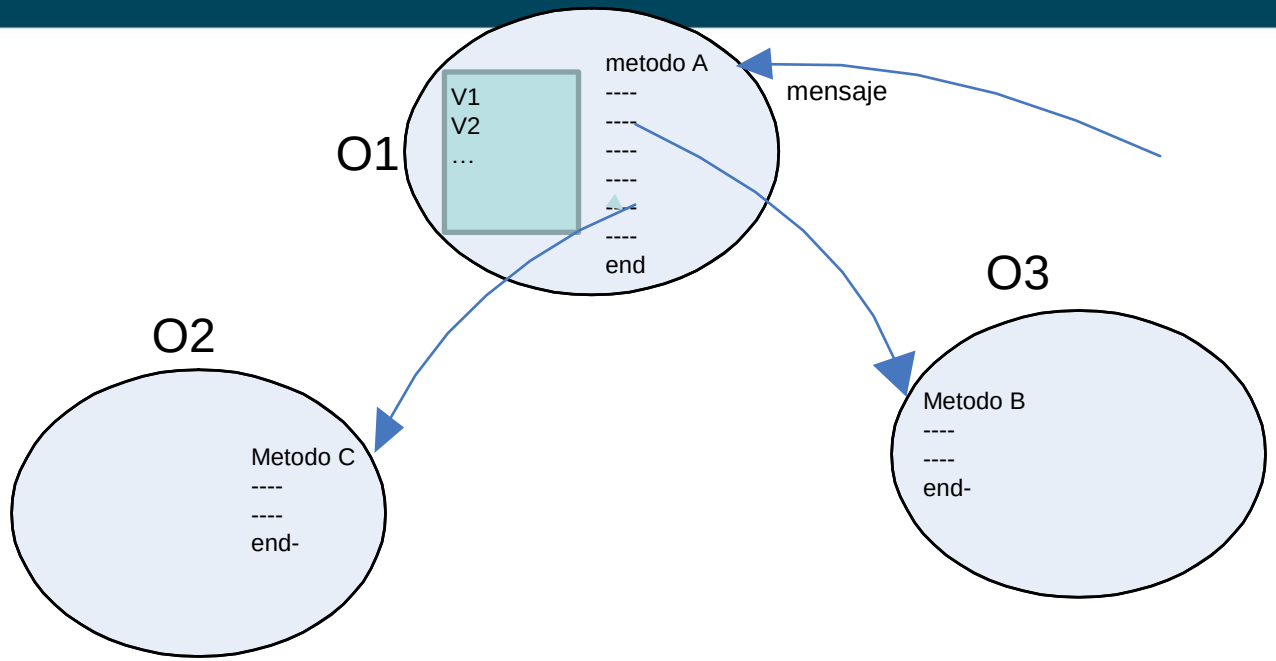
- El estado interno de un objeto determina su *conocimiento*.

- El estado interno está dado por:
  - Propiedades basicas (intrinsecas) 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. (Cual 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 basicos
- 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

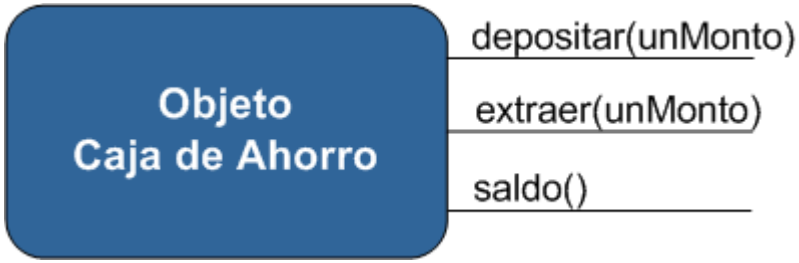
Materias es una colección (array?) de objetos Materia

- Objeto Materia:

  v.i: nombre, año, semestre, correlativas

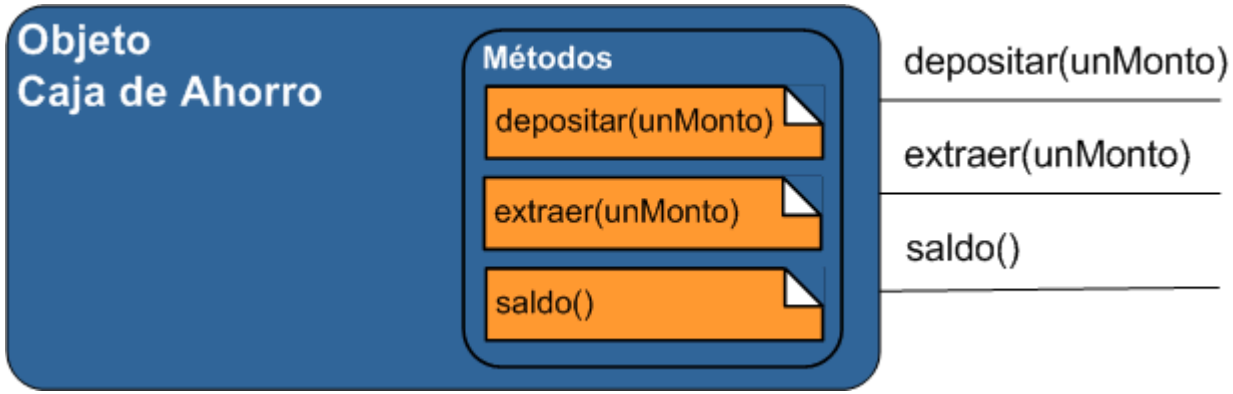
Comportamiento

- 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:



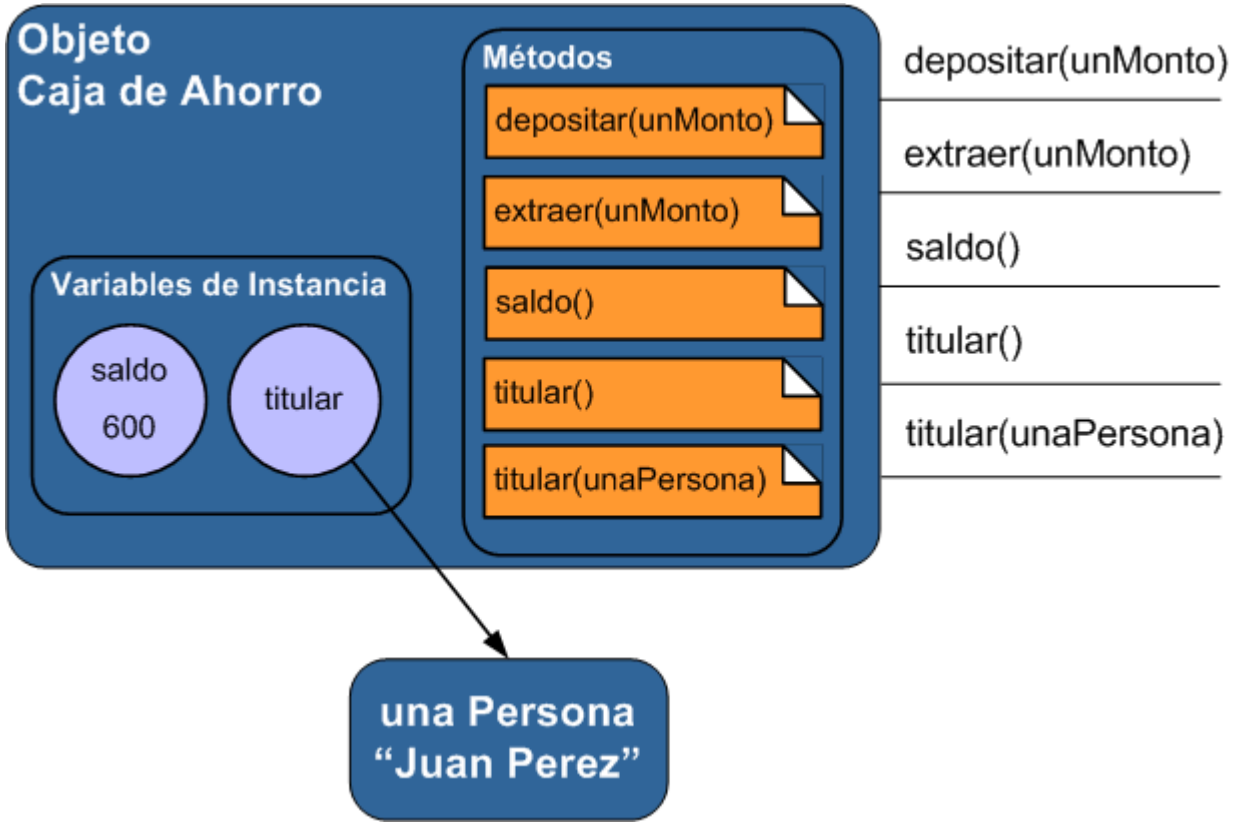
Comportamiento – implementación

- La **realizacion** 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



Observese 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

figura.rotar

producto.precio

- Tanto figura como producto 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 deberia 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
- Que 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 – Depositar en Cuenta Bancaria

```
..... depositar (Integer unMonto)

{
  saldo = saldo + unMonto
  return ...
 }
```

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”)

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.

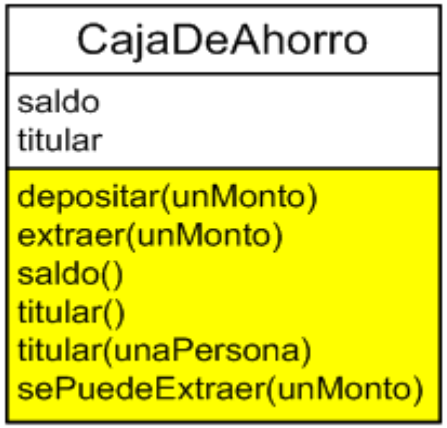
Clases e instancias

- 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:

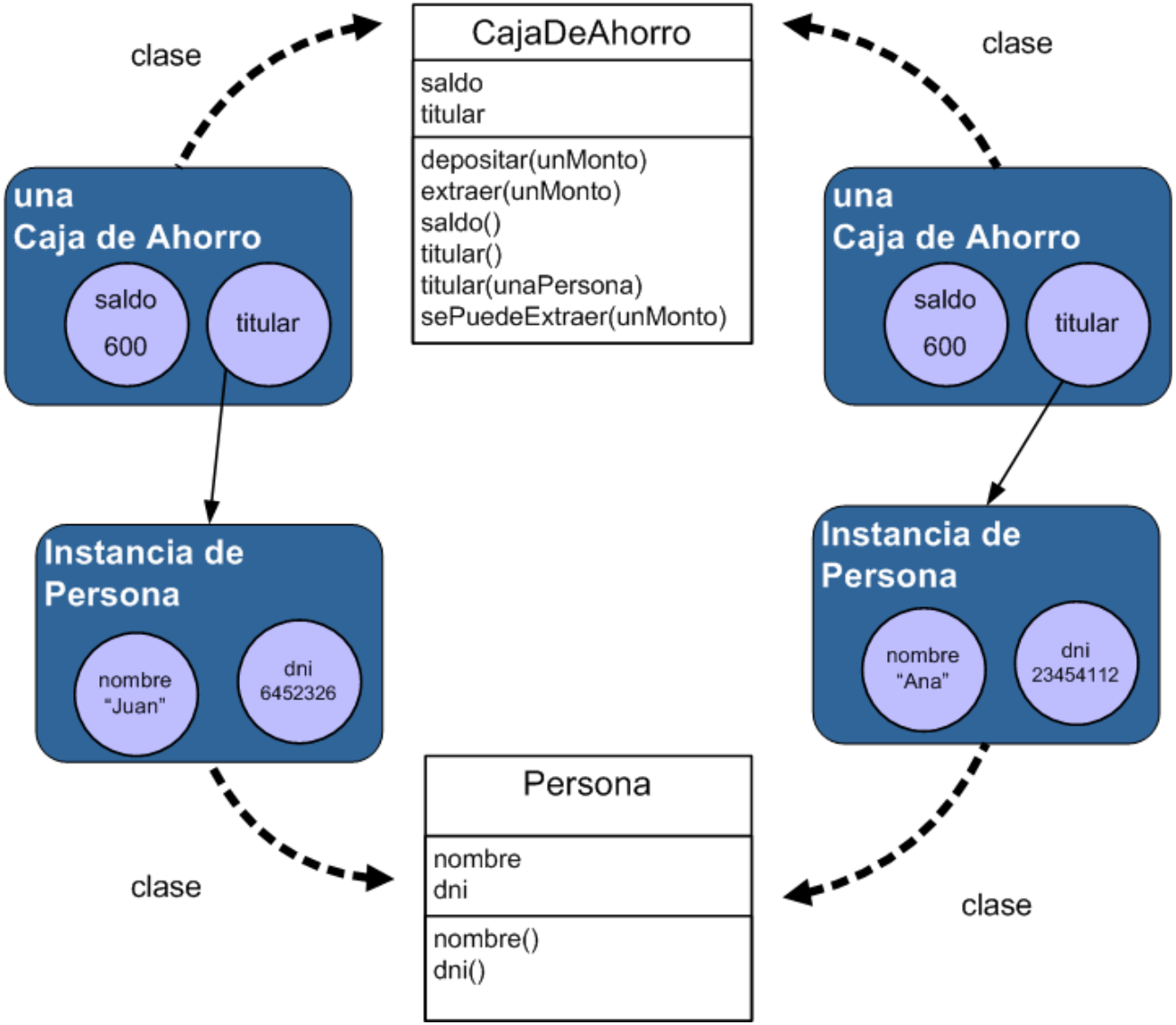
Variables de Instancia
Los nombre de las v.i. se escriben en minúsculas y sin espacios



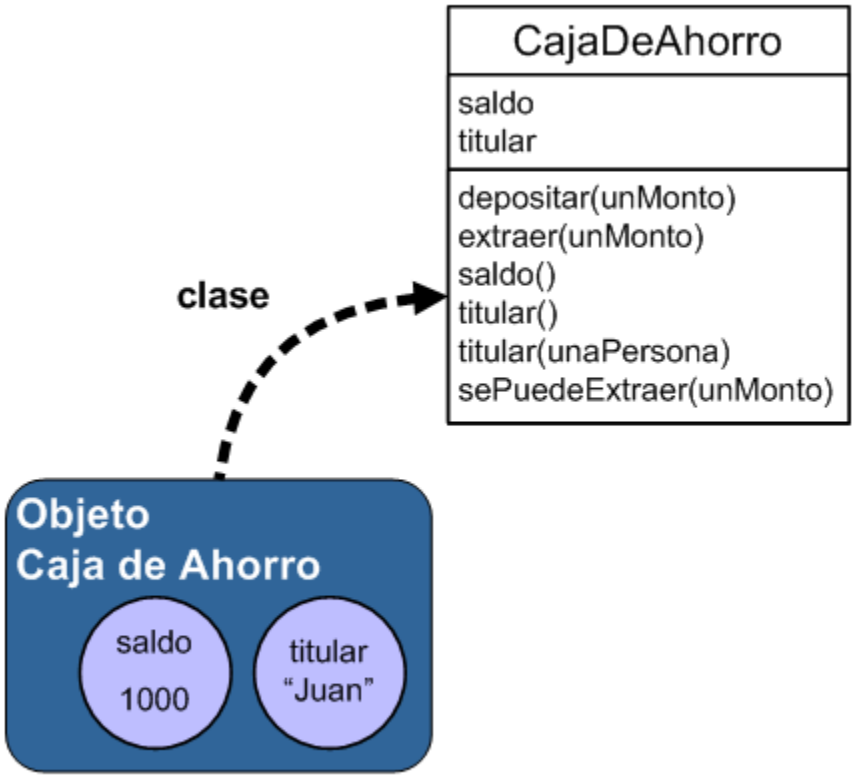
Nombre de la Clase
Comienzan con mayúscula y no posee espacios

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

Ejemplo de clases e instancias

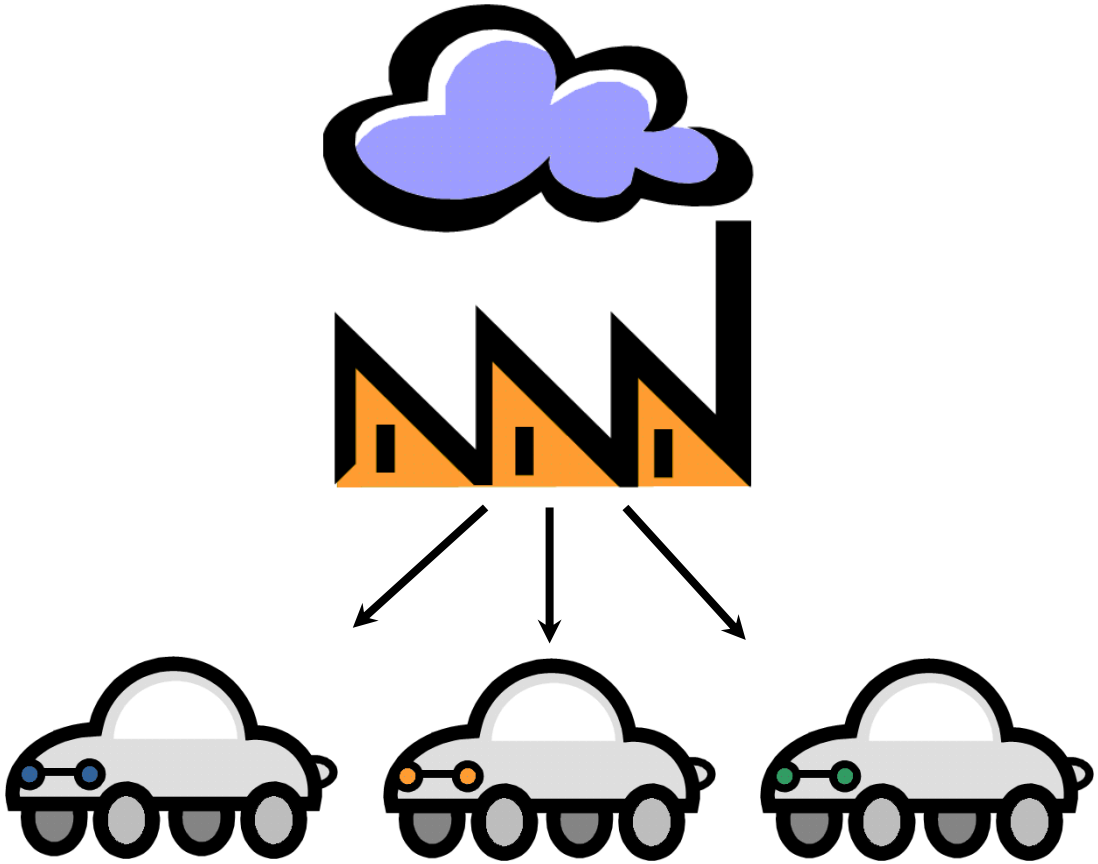


Envío de mensajes con clases



Creación de Objetos

- ¿Cómo creamos nuevos objetos?
- Instanciación



Creación de Objetos

- Comúnmente se utiliza la palabra reservada ***new*** para instanciar nuevos objetos.
- Quien crea objetos? Cuando los crea?

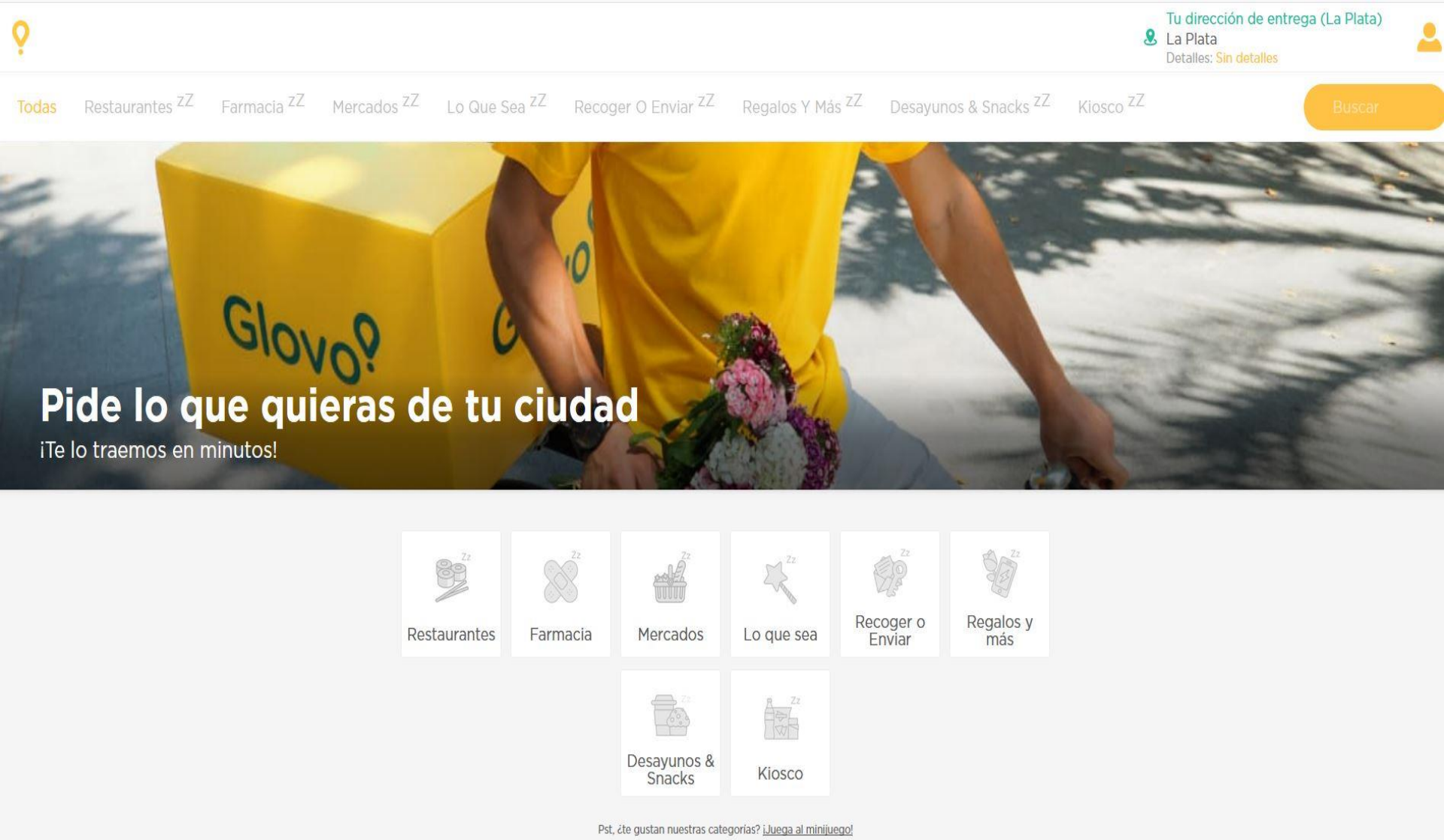
Instanciación

- 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).

Inicialización

- 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

Ejemplo: Gloovo



Objetivos del Ejemplo

- Repasar los conceptos basicos usando un ejemplo realista
- Vamos a analizar el diseño sin preocuparnos por COMO llegamos a el (por ahora).

Se trata de describir con objetos la funcionalidad mas importante de un sistema del tipo Rapi, Glovo o PedidosYa, que permite que cualquier usuario registrado (cliente) pida un producto en un comercio registrado y alguien (otros usuarios especiales que se postulan como “Gloovers") le lleve el producto a la casa.

Con un conjunto de opciones desplegables el "cliente" elige producto (o producto y negocio en el que lo quiere comprar) y registra un pedido. En ese momento sabe el precio que pagará tanto por el producto como por el envío. El sistema todavía no puede estimar un tiempo de entrega. El “sistema” recibe el pedido y lo postea entre los usuarios “Gloovers". Cuando alguno de ellos lo “toma”, el pedido se asigna al "acarreador“ (un gloover) y se comunica al negocio que provee el producto para que lo prepare. El Gloover va al negocio y retira el producto. Utiliza la aplicación para confirmar que retiro el producto. Se dirige a la dirección de entrega y entrega el producto. Utiliza la aplicación para confirmar que lo entregó.

Cuando el cliente recibe el pedido, utiliza la aplicación para confirmar la recepción. En ese momento, se carga el costo a su medio de pago, se para al negocio, y se paga al "acarreador". Adicionalmente el cliente puede calificar al gloover con un puntaje y un comentario

Contexto del Ejemplo y restricciones

- No vamos a preocuparnos (ahora) por el diseño de las interfases o los formularios.
- No vamos a preocuparnos por la comunicacion entre los usuarios (desde telefonos, tabletas, computadoras, etc) y la aplicacion
- Vamos a ignorar el “almacenamiento” de los datos (en archivos, bases de datos, centralizado, distribuido, etc)

- Sin embargo podemos decir:
  - Es correcto asumir que esos aspectos no implicaran cambio alguno en el sistema (nuevas interfases por ejemplo, nuevos dispositivos, etc)
  - La “comunicacion” es transparente al software (casi siempre)