PROGRAMACIÓN III

2021

CLASE 1: POO. Primeros conceptos

- Ideas básicas
- Una nueva forma de pensar
- Paradigmas
- Paradigma estructurado vs. paradigma orientado a objetos
- El Objeto
- La Clase (anticipo)
- Objetos y Clases (anticipo)
- Instanciación de Clases, variables, referencias.
- Componentes de un objeto: Estado, Comportamiento,
- Interacción entre objetos: mensajes.
- Métodos. Pasaje de parámetros, retorno.
- Clases: Constructores, Instanciación de clases
- Manipulación de objetos
- El método main()
- Abstracción de la realidad al modelar con objetos
- UML. Diagrama de Clases
- Ejemplo de un programa en Java – Implementación TDA Pila en objetos

IDEAS BÁSICAS

La programación orientada a objetos es una metodología que descansa en el concepto de objeto para imponer la estructura modular de los programas. Permite **comprender el dominio del problema a resolver, al intentar construir un modelo del mundo real que envuelve nuestro sistema.** Es decir, la representación de este mundo mediante la identificación de los objetos que constituyen el vocabulario del dominio del problema, su organización y la representación de sus responsabilidades.

IDEAS BÁSICAS

La idea de manejar objetos reales como contenedores de estados y comportamientos, es mucho más atractiva desde el punto de vista del usuario. De esta manera no tendrá que batallar con **construcciones** orientadas al computador, sino que podrá manejar objetos (y operaciones) **que se asemejen más a sus equivalentes en el mundo real.** La elevación del nivel de abstracción es sin duda un objetivo deseable.

IDEAS BÁSICAS

Las técnicas orientadas a objetos usan un mismo modelo conceptual para el análisis, el diseño y la programación. La transición desde el análisis al diseño es tan natural que es difícil especificar donde comienza uno y donde acaba el otro.

UNA NUEVA FORMA DE PENSAR

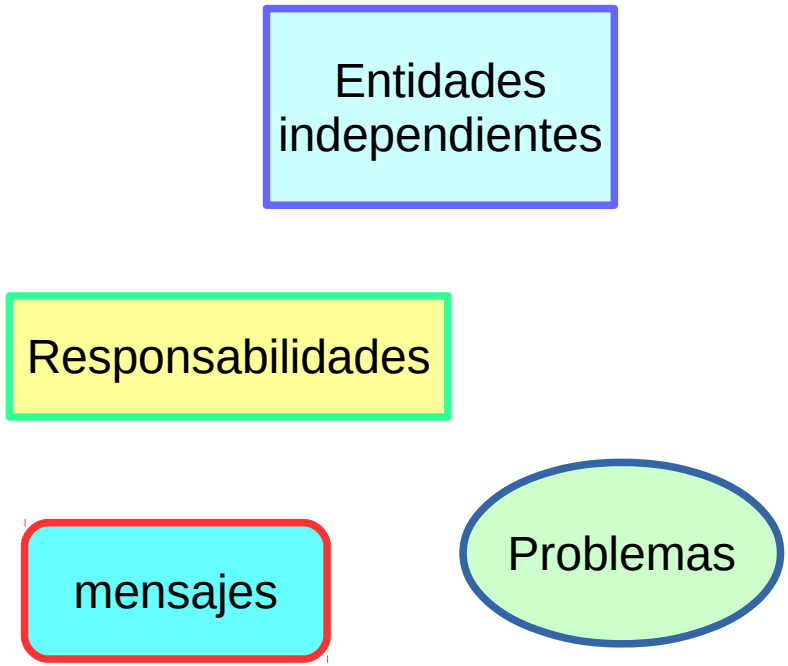
La P.O.O. hace referencia a una **nueva forma de pensar** (más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación) acerca del proceso de descomposición de problemas y de desarrollo de soluciones de programación.

Tradicionalmente, la forma de enfrentarse a la complejidad de la programación ha sido descomponer el problema objeto de resolución en **subproblemas** y más subproblemas hasta llegar a acciones muy simples y fáciles de codificar (*paradigma estructurado - funcional*)

UNA NUEVA FORMA DE PENSAR

En la **P.O.O.** se buscan (o diseñan) **entidades independientes que colaboren** en la resolución de un problema. Cada una de estas entidades tendrá un conjunto de datos propios y responsabilidades designadas (cada entidad es responsable de ciertas tareas).

Para resolver un problema, se utilizan un conjunto de estas entidades, las cuales **interactúan entre si**, enviándose **mensajes entre ellas** y proporcionando en cada mensaje la información necesaria para resolver la tarea y devolver una respuesta a quien le solicitó.



PARADIGMA

El filósofo y científico Thomas Kuhn dio a la palabra **paradigma** su significado contemporáneo, lo define de la siguiente manera:

- lo que se debe observar y escrutar (Observar o examinar algo o a alguien con mucha atención y minuciosidad.)
- el tipo de interrogantes que se supone hay que formular para hallar respuestas en relación al objetivo
- cómo deben estructurarse estos interrogantes
- cómo deben interpretarse los resultados de la investigación científica

"Considero a los paradigmas como realizaciones científicas universalmente reconocidas que, durante cierto tiempo, proporcionan modelos de problemas y soluciones a una comunidad científica"

Thomas Kuhn

PARADIGMA

*La programación Orientada a Objetos (POO) es un **paradigma de programación** que usa objetos y sus interacciones para diseñar aplicaciones y programas. Está basado en varias técnicas, como por ejemplo herencia, abstracción, polimorfismo y encapsulamiento.*

En síntesis ¿Qué es un paradigma?

Conjunto de elementos y reglas. Brinda un marco para el diseño y la construcción de Programas.

PARADIGMA ESTRUCTURADO VS. PARADIGMA ORIENTADO A OBJETOS

Paradigma Estructurado

- ✔ Sistemas contienen datos y algoritmos.
- ✔ Los programas manipulan los datos.
- ✔ Se construyen en base a
  - ✔ Asignación.
  - ✔ Secuencia.
  - ✔ Repetición.
  - ✔ Condicional.
- ✔ Los programas están organizados por:
  - ✔ Descomposición funcional.
  - ✔ Flujo de Datos.
  - ✔ Módulos.

Paradigma Orientado a Objetos

- ✔ Los sistemas están compuestos por un conjunto de objetos.
- ✔ Los objetos son responsables de llevar a cabo ciertas acciones.
- ✔ Los objetos colaboran para llevar a cabo sus responsabilidades.
- ✔ Los programas están organizados en base a clases y jerarquías de herencia.
- ✔ Principios del paradigma OO según Alan Kay (creador del Smalltalk)
  - ✔ Todo es un objeto.
  - ✔ Los objetos se comunican enviando y recibiendo mensajes.
  - ✔ Los objetos tienen su propia memoria (en términos de objetos).

EL OBJETO

Un **objeto** en POO *representa* alguna entidad de la vida real, es decir, alguno de los objetos que pertenecen al negocio con que estamos trabajando o al problema con el que nos estamos enfrentando, y con los que podemos interactuar.

Auto
patente modelo marca color ….
arrancar() detener() acelerar() frenar() ….

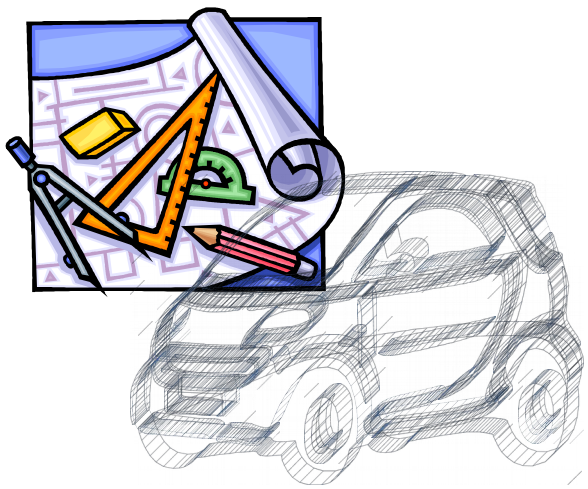
Objeto es todo aquello que pueda ser identificable dentro de una especificación de requerimientos o problema, y posea:

- **Características:** una manera de representar sus datos.
- **Comportamiento:** lo que puede hacer (correr, saltar, volar, etc).
- **Interacción/comunicación:** con otros objetos por medio de sus métodos

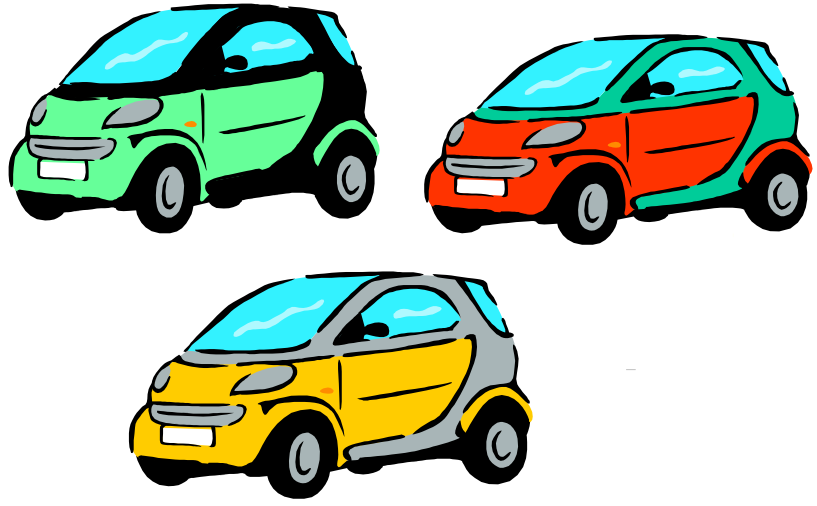
LA CLASE

A través del estudio de los objetos se adquiere el conocimiento necesario para, mediante la abstracción y la generalización, agruparlos según sus características en conjuntos. Estos conjuntos determinan las **clases** de objetos con las que estamos trabajando.

En general, primero existen los objetos; luego aparecen las clases en función de la solución que estemos buscando.



CLASE



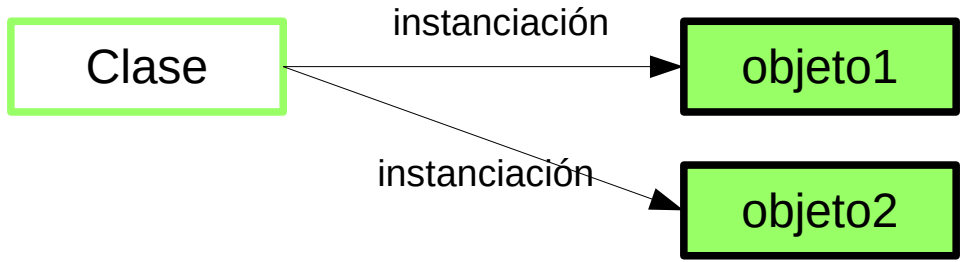
OBJETOS

OBJETOS Y CLASE (anticipo)

- Una clase es una definición abstracta de un objeto
  - Define la estructura y el comportamiento compartidos por los objetos
  - Sirve como modelo para la creación de objetos
- Los objetos pueden ser agrupados en clases

En algún punto, la clase es al objeto como el struct a la variable de ese tipo.-

El mecanismo para la **creación** de objetos a partir de la clase (que define sus características y comportamiento) se llama **instanciación** de la clase



INSTANCIACIÓN DE CLASES, VARIABLES, REFERENCIAS

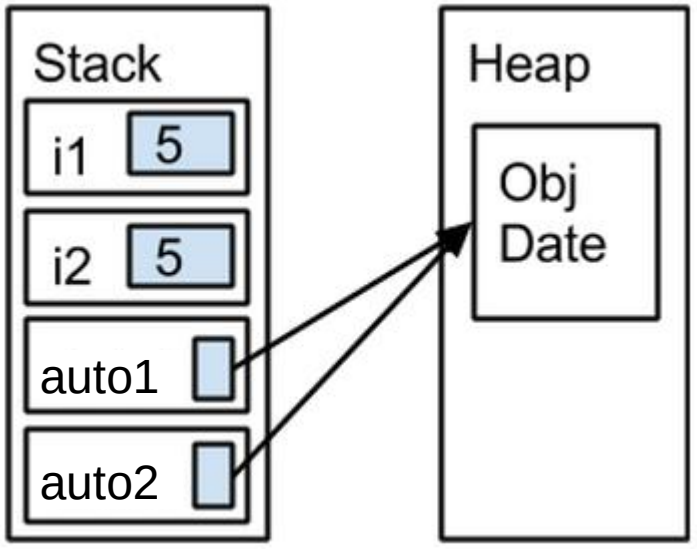
Para instanciar una clase y así crear un nuevo objeto, se utiliza el operador **new(...)**

```
Auto auto1 = new Auto();
```

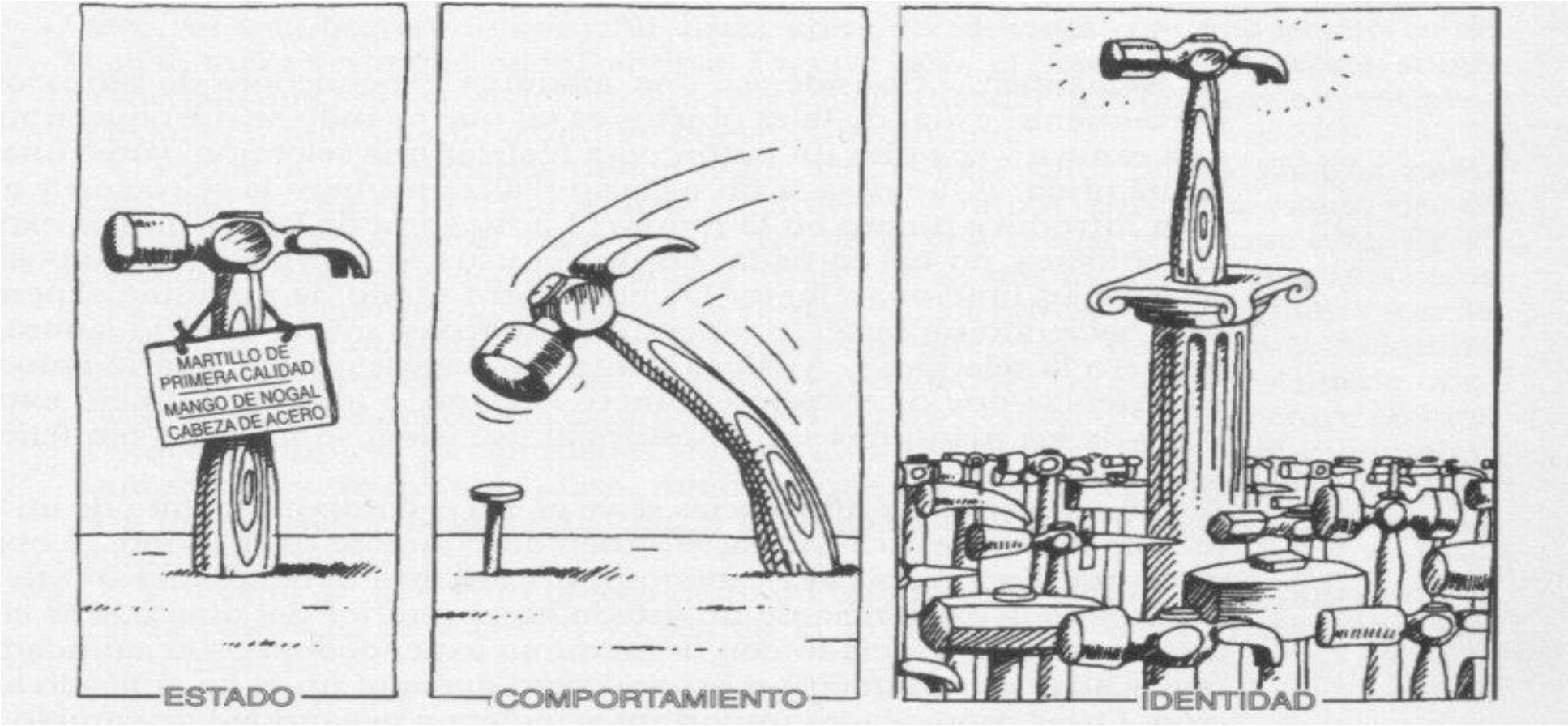
clase | variable | operador | constructor

El operador **new (...)** crea un nuevo objeto y lo almacena en el HEAP (espacio de almacenamiento para los objetos) y retorna una referencia al objeto creado. Esta referencia se almacena en una variable del tipo (de la clase)

Si bien no es lo mismo, ayuda pensar en el concepto de puntero (C++). En JAVA no existen punteros.



COMPONENTES DE UN OBJETO

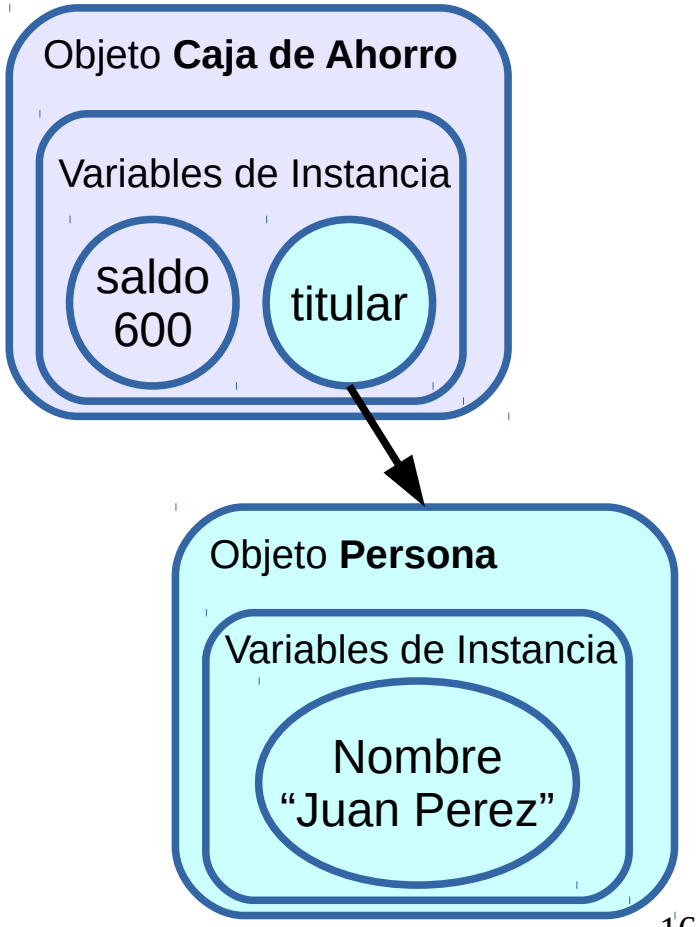


Un objeto tienen estado, exhibe algún comportamiento bien definido, tiene una identidad única.

EL ESTADO DE UN OBJETO

El estado de un objeto se refiere al conjunto de los valores de sus atributos en un instante de tiempo dado.

- Está compuesto por las **variables de instancia** del objeto (atributos).
- Las **V.I.** pueden hacer referencia a:
  - Propiedades intrínsecas del objeto.
  - Otros objetos con los cuales puede colaborar para llevar a cabo sus responsabilidades.
- Es conveniente que sea privado del objeto. Ningún otro objeto debería accederlo.



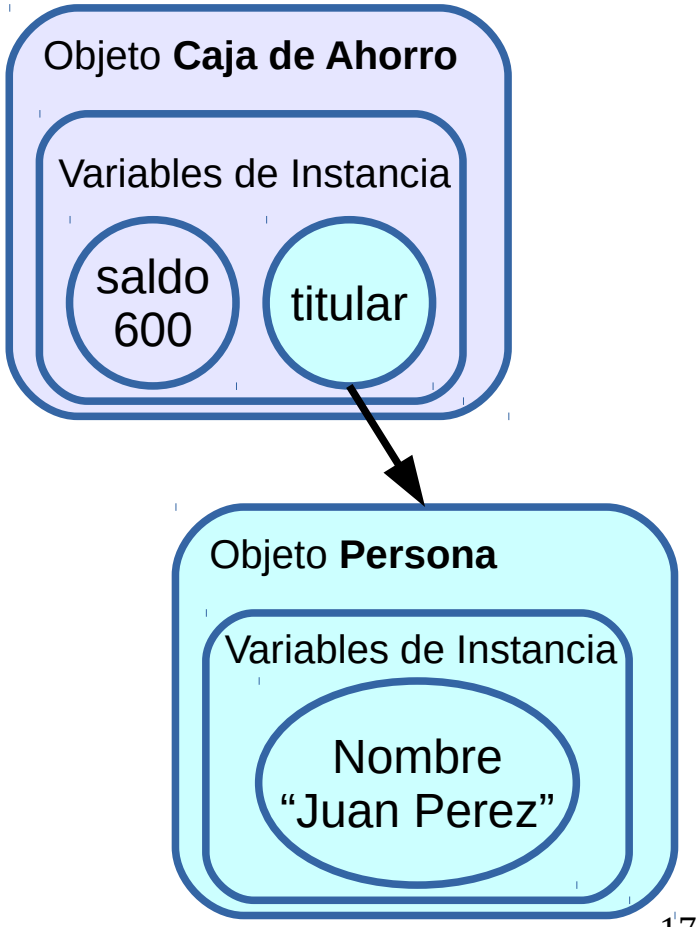
EL ESTADO DE UN OBJETO – VARIABLES DE INSTANCIA

Variable

Es unidad básica de almacenamiento. En el Lenguaje Java, cada variable tiene un tipo de dato, un identificador y un ámbito.

Variables de Instancia o atributos

Los atributos son las características individuales que diferencian un objeto de otro. Los atributos se guardan en variables denominadas de instancia y cada objeto particular puede tener valores distintos para estas variables.



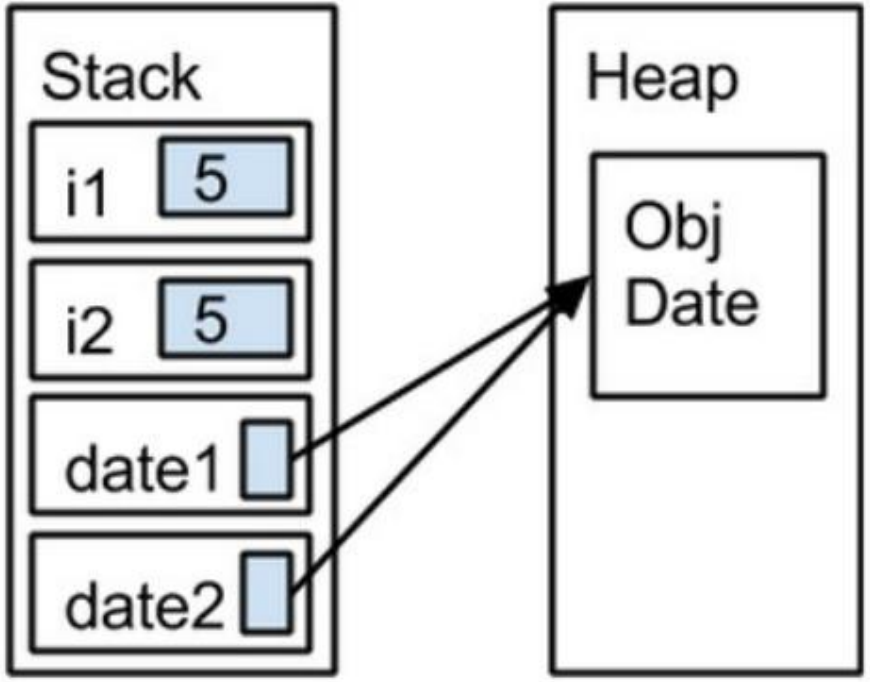
TIPOS DE DATOS DE LAS VARIABLES

Las variables pueden contener tipos de datos primitivos o referencia a objetos

Tipos de datos primitivos: entero, carácter, booleano...

Referencia a Objetos: las variables de tipo no primitivo almacenan una referencia a un objeto, no contienen al objeto. Los objetos se almacenan en una pila de objetos llamada Heap.

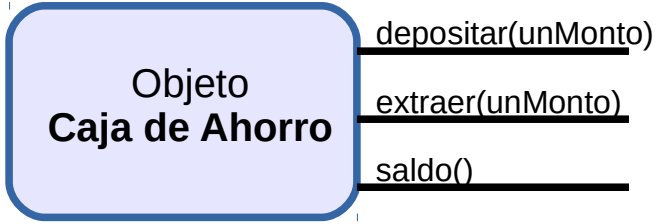
Si bien no es lo mismo, ayuda pensar en el concepto de puntero (C++).



EL COMPORTAMIENTO DE UN OBJETO

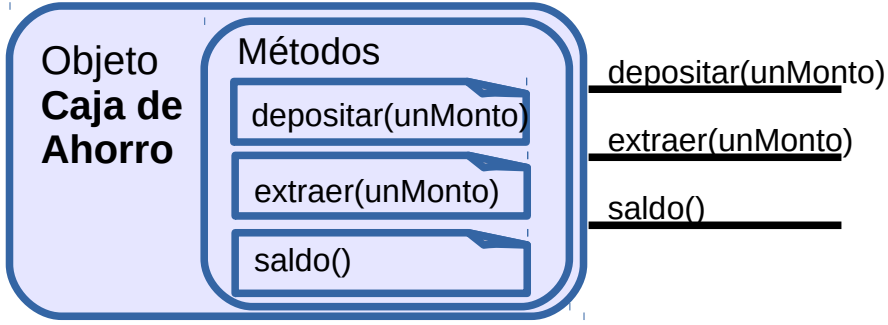
¿Qué hace un objeto?

Un objeto se define en términos de su comportamiento.
El comportamiento indica qué sabe hacer el objeto (determina cuales son sus responsabilidades)
Se especifica a través del conjunto de mensajes que el objeto sabe responder: protocolo.

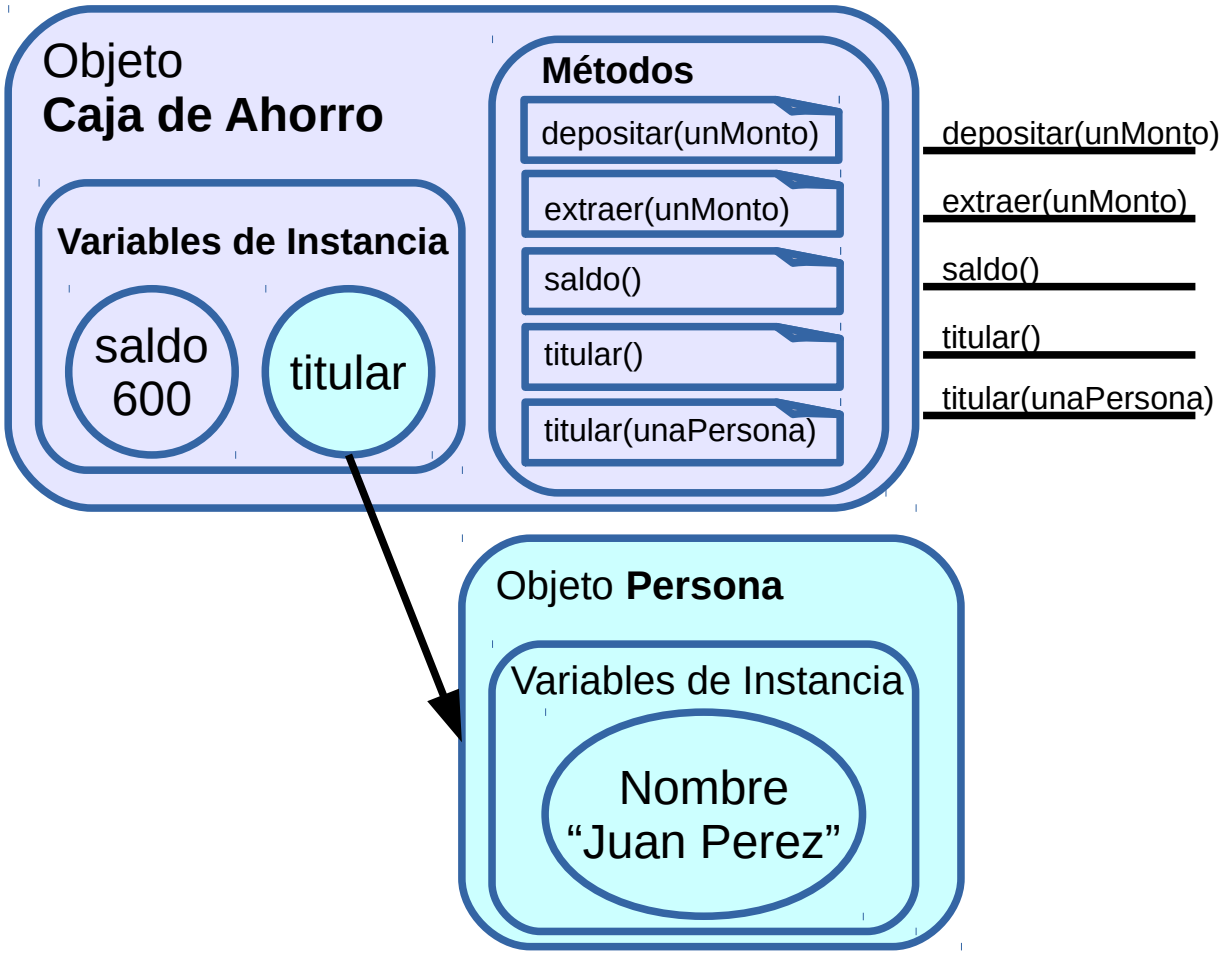


¿Cómo lo hace?

La implementación de un objeto se especifica a través de un conjunto de métodos.
Cuando un objeto recibe un mensaje responde activando el método asociado.
Es privado del objeto: el que envía el mensaje delega en el receptor la manera de resolverlo, sin importarle cómo lo resuelve.



Estado y Comportamiento de un Objeto



INTERACCIÓN ENTRE OBJETOS USANDO MENSAJES

El **comportamiento** de un objeto está definido en términos de los **mensajes que éste entiende.**

Para poder enviarle un mensaje a un objeto, primero 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.

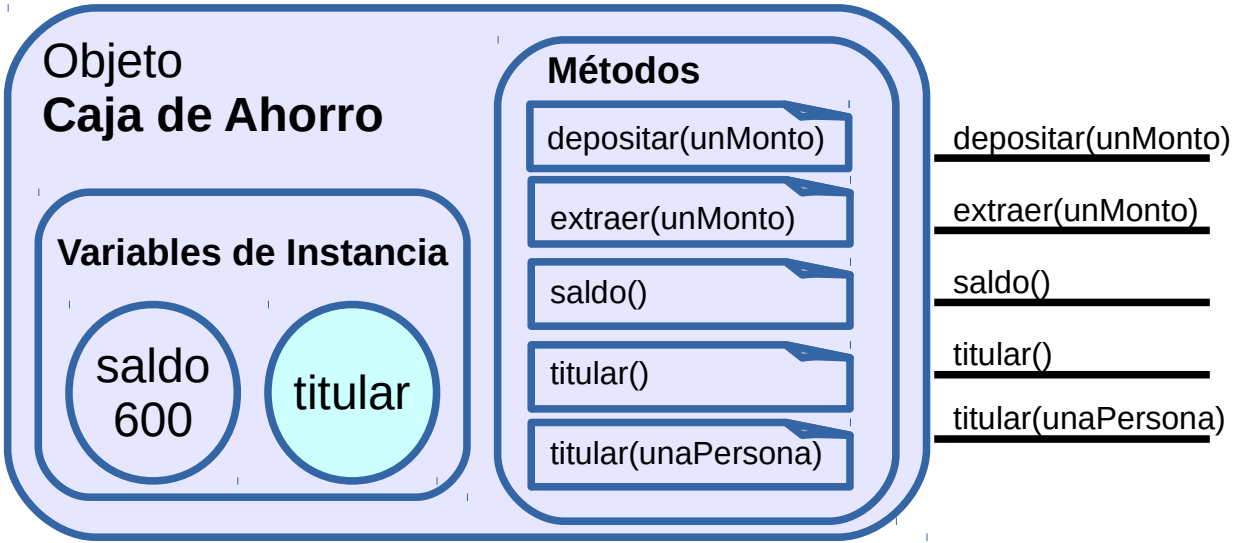
Se especifica con el nombre correspondiente al protocolo del objeto receptor.

Se indica cuales son los parámetros, es decir, la información necesaria para resolver el mensaje.

Se utilizará la siguiente sintaxis:

```
<objeto receptor>.<nombre del mensaje>(<parámetros>);
```

EJEMPLO DE MENSAJE



Ejemplo empleando la sintaxis propuesta:

Decirle a una cuenta bancaria que deposite $100 se escribe como:

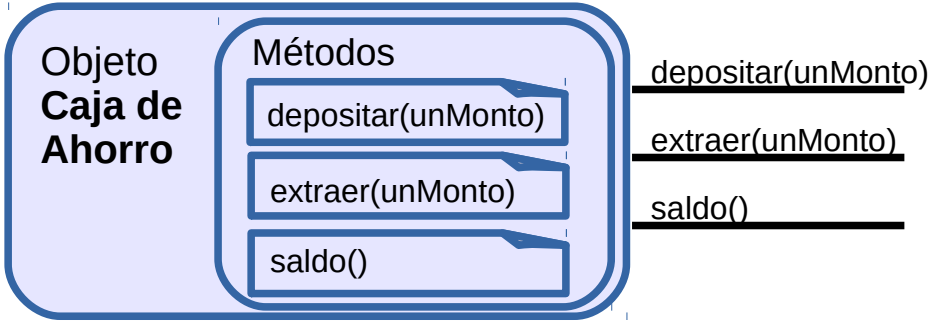
```
CajaDeAhorro unaCuenta = new CajaDeAhorro();
unaCuenta.depositar(100);
```

MÉTODOS

Los métodos son la contraparte funcional del mensaje. Expresa la forma de llevar a cabo la semántica propia de un mensaje particular.

Un método puede realizar básicamente 3 cosas:

- Modificar el estado interno del objeto receptor.
- Colaborar con otros objetos.
- Retornar y terminar.




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

```
public double extraer (double monto)
{
      This.saldo = this.saldo – monto;
       return saldo;
}
```

```
public double saldo()
{
    return this.saldo;
}
```

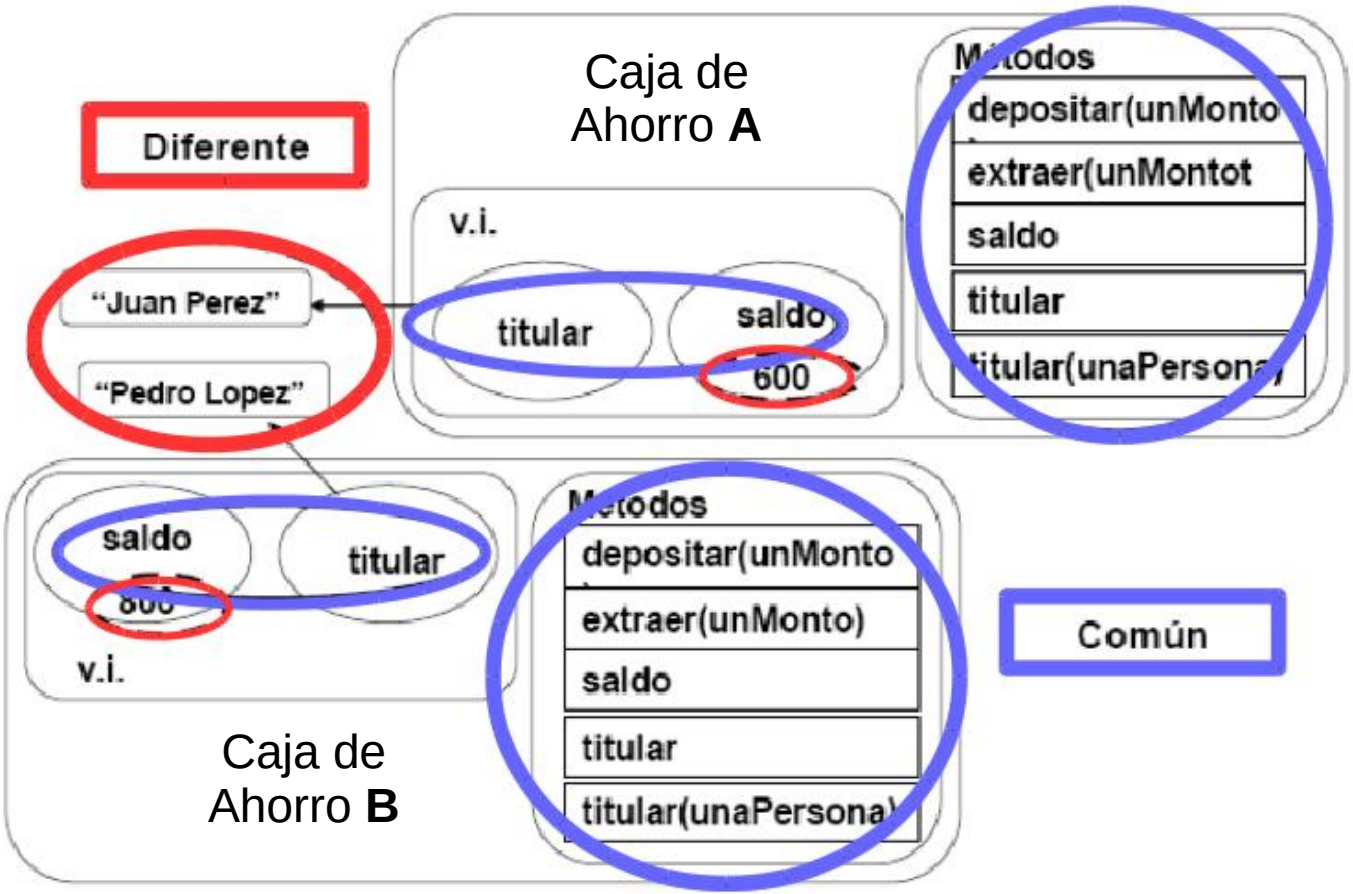
PARÁMETROS, PASAJE DE PARÁMETROS, RETORNO

En el Lenguaje de Programación Java, el **pasaje de parámetros es por valor.**

Tener en cuenta que en el caso de parámetros de tipo Objeto, **el valor que contiene es una referencia a un Objeto** (como cualquier variable no primitiva). Por lo tanto, teniendo esa referencia se puede **alterar el estado** del objeto referenciado (por el parámetro formal).

En el momento de la invocación, el parámetro formal copia el contenido del parámetro real (pasaje de parámetros por valor), pero lo que está copiando es la referencia del objeto que contiene el parámetro real). O sea, cualquier alteración (dentro del método) del estado del objeto referenciado por el parámetro formal impacta sobre el objeto.

ASPECTOS COMUNES A LOS OBJETOS



Comportamiento común entre objetos

¿Cuántos objetos Caja de Ahorro habrá?

¿Es necesario especificar el comportamiento de cada caja de Ahorro? ¿O el comportamiento debería ser común a todas?

¿Qué cosas son comunes a todas las cajas de ahorro y qué cosas son particulares a cada una?

Entonces... ¿cómo representar este comportamiento común, de manera que cada caja de ahorro pueda reutilizarlo?


CLASES !

Clases

Una clase es una descripción de un grupo de objetos con:

- Propiedades en común (atributos)
- Comportamiento similar (operaciones)
- La misma forma de relacionarse con otros objetos (relaciones)
- Una semántica en común (significan lo mismo)

Una clase es una abstracción que:

- Enfatiza las características relevantes
- Suprime otras características (simplificación)

Un objeto es una instancia de una clase

Una clase es una definición abstracta de un objeto

- Define la estructura y el comportamiento compartidos por los objetos
- Sirve como modelo para la creación de objetos

Los objetos pueden ser agrupados en clases

Clases e Instancias (objetos)

Después de identificar los requerimientos de una aplicación, debemos identificar las clases y los objetos necesarios para construir la aplicación, así que vamos a ver qué son clases y objetos.

Clase
Concepto: Una clase es la abstracción de las propiedades y operaciones o acciones que ejecutan los objetos del mundo real.

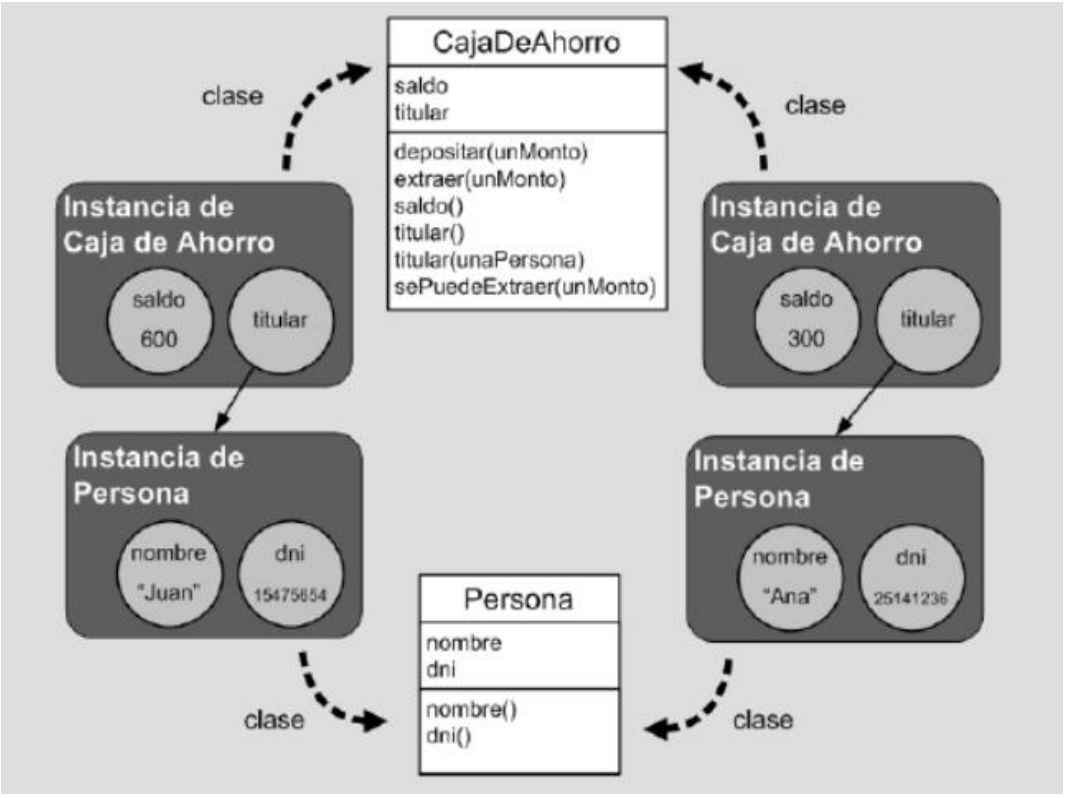
en términos coloquiales
la clase
es como si fuera una plantilla
que sirve para
crear los objetos

Objeto
Concepto: Los objetos son instancias de una clase

es decir, las **unidades** de una **clase específica** que existen en el mundo real

Una similitud es la siguiente:
La clase es como un molde para hacer tortas y las tortas hechas con el molde son los objetos

Clases e Instancias (objetos)



Las clases cumplen tres roles:

- Agrupar el comportamiento común
- Definir la “forma”
- Crear las instancias

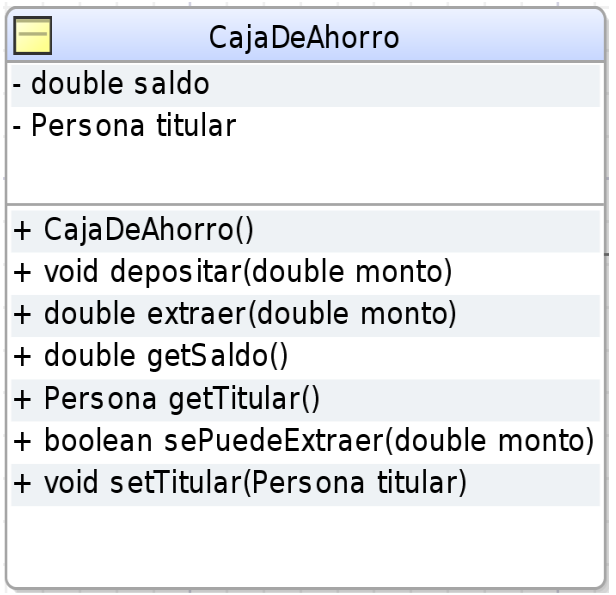
En consecuencia, todas las instancias de una clase se comportan de la misma manera. Cada instancia mantendrá su propio estado interno.

Cada instancia tiene una copia de las variables de instancia y de los métodos declarados en la clase.

Especificación de Clases

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

Nombre de Clase:
Comienza con mayúscula y no posee espacios



Atributos (variables de instancia):
Se escriben en minúsculas y sin espacios

Métodos:
Para cada uno se debe especificar como mínimo el nombre, parámetros y retorno

El constructor

- Se utiliza para realizar inicializaciones más complejas.
- Bloque de sentencias que se pueden utilizar para inicializar un objeto antes de que se devuelva una referencia mediante new.
- Tiene el mismo nombre que la clase que inicializan
- Puede admitir cero o más parámetros.
- No es un método y por lo tanto no tiene tipo de retorno.
- Se invoca después de que las variables de instancia del objeto de nueva creación tenga asignado sus valores iniciales por defecto y después de que sus inicializadores explícitos se hayan asignado.

Constructores

```java
public class Persona
{
    private String nombre;

    public Persona(String nombre)
    {
        this.nombre = nombre;
    }

    public void setNombre(String nombre)
    {
        this.nombre = nombre;
    }

    public String getNombre()
    {
        return nombre;
    }
}
```

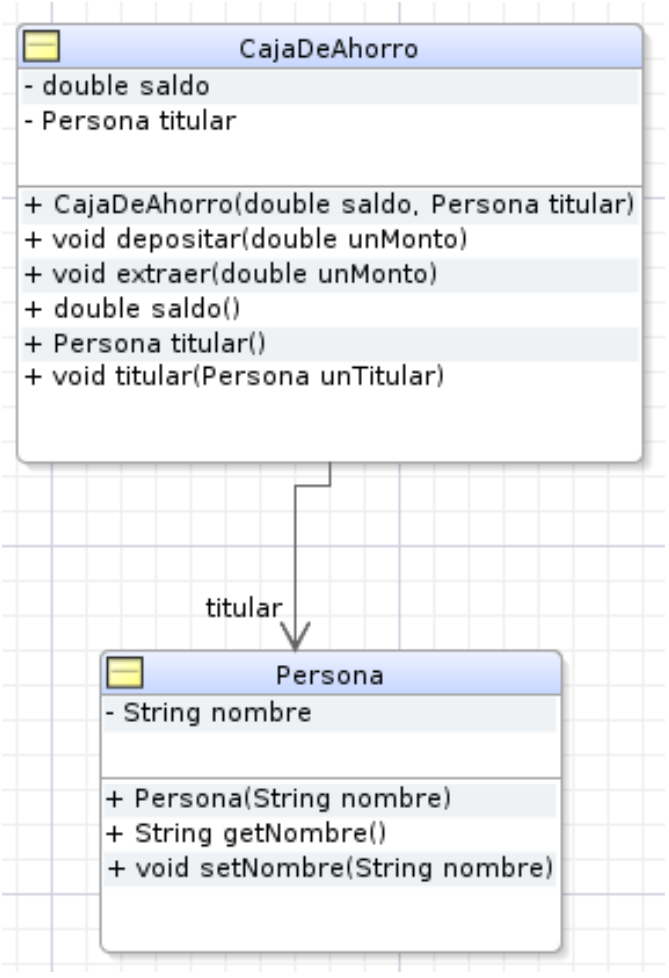
```java
public class CajaDeAhorro
{
    private double saldo;
    private Persona titular;

    public CajaDeAhorro(double saldo, Persona titular)
    {
        this.saldo = saldo;
        this.titular = titular;
    }

    public void depositar(double unMonto)
    {
        double nuevoSaldo;
        nuevoSaldo = this.saldo + unMonto;
        saldo = nuevoSaldo;
    }

    public void extraer(double unMonto)
    {
        this.saldo = this.saldo - unMonto;
    }
```

Cómo se instancia una Clase ?

- Para instanciar una clase, es decir, para crear un objeto de una clase, se usa el operador **new**.
- La creación e inicialización de un objeto involucra los siguientes pasos:
  - Se aloca espacio en memoria para la variable y para el objeto.
  - Se inicializan los atributos del objeto con los valores por defecto.
  - Se ejecuta el constructor (parecido a un método que tienen el mismo nombre de la clase)
  - Se asigna la referencia del nuevo objeto a la variable.

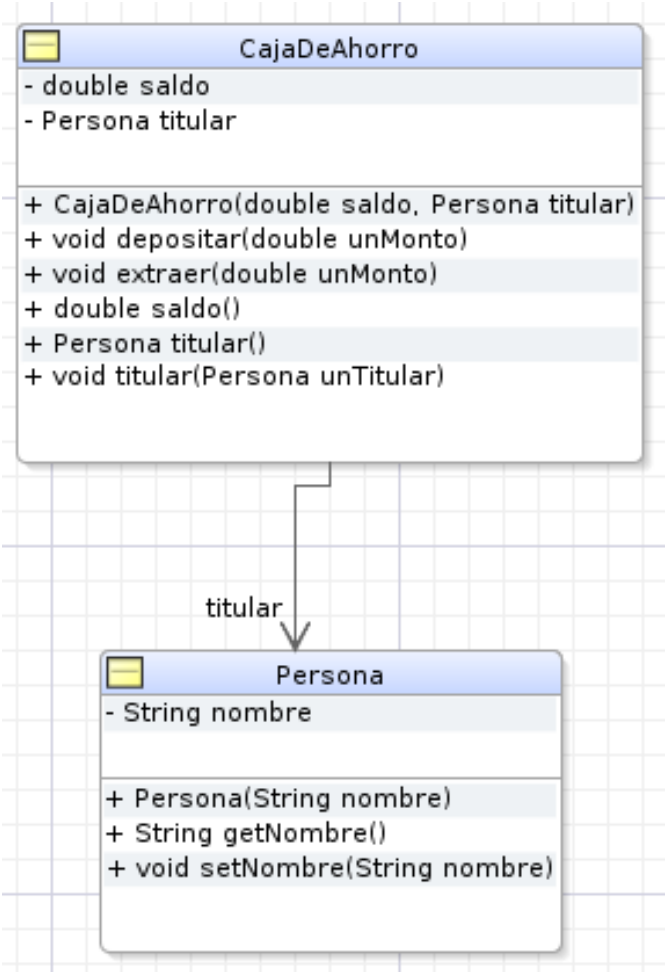
```
CajaDeAhorro miCaja = new CajaDeAhorro(saldo, titular);
```

Especificación de Clases




```java
public class Persona
{
    private String nombre;

    public Persona(String nombre)
    {
        this.nombre = nombre;
    }

    public void setNombre(String nombre)
    {
        this.nombre = nombre;
    }

    public String getNombre()
    {
        return nombre;
    }
}
```

Especificación de Clases




```java
public class CajaDeAhorro
{
    private double saldo;
    private Persona titular;

    public CajaDeAhorro(double saldo, Persona titular)
    {
        this.saldo = saldo;
        this.titular = titular;
    }

    public void depositar(double unMonto)
    {
        double nuevoSaldo;
        nuevoSaldo = this.saldo + unMonto;
        saldo = nuevoSaldo;
    }

    public void extraer(double unMonto)
    {
        this.saldo = this.saldo - unMonto;
    }

    public double saldo()
    {
        return this.saldo;
    }

    public Persona titular()
    {
        return this.titular;
    }

    public void titular(Persona unTitular)
    {
        this.titular = unTitular;
    }
}
```

Cómo manipular Objetos?

Una vez que se ha creado un objeto, seguramente necesitemos usarlo: cambiar su estado, obtener información o ejecutar alguna acción. Para poder hacerlo se necesita:

- conocer la variable referencia
- utilizar el operador "."

Invocar sus métodos públicos

```
Cuenta miCuenta = new Cuenta();
miCuenta.setSaldo(1500);
double sal = miCuenta.getSaldo();
```

Cuenta.java

```
public class Cuenta {
 private int cuentaId;
 private double saldo=0;
 private char tipo;
 public double getSaldo(){
      return saldo;
 }
 public void setSaldo(double s){
      saldo = s;
 }
}
```

El método main()

Para que una clase pueda ejecutarse debe contener el siguiente método:

```
public static void main(String args[]){. . .}
```

calificadores obligatorios

Acepta objetos de tipo String, aunque no es obligatorio enviarlos.

Dada una clase Cajero, que puede ejecutarse:

```
public class Cajero {

 public static void main(String[] args) {
    System.out.println("Buen dia !!!"+args[0]+" "+args[1]);
 }

}
```

Si ejecutamos desde la línea de comandos la clase Cajero:

```
C:\java Cajero María Jeréz
```

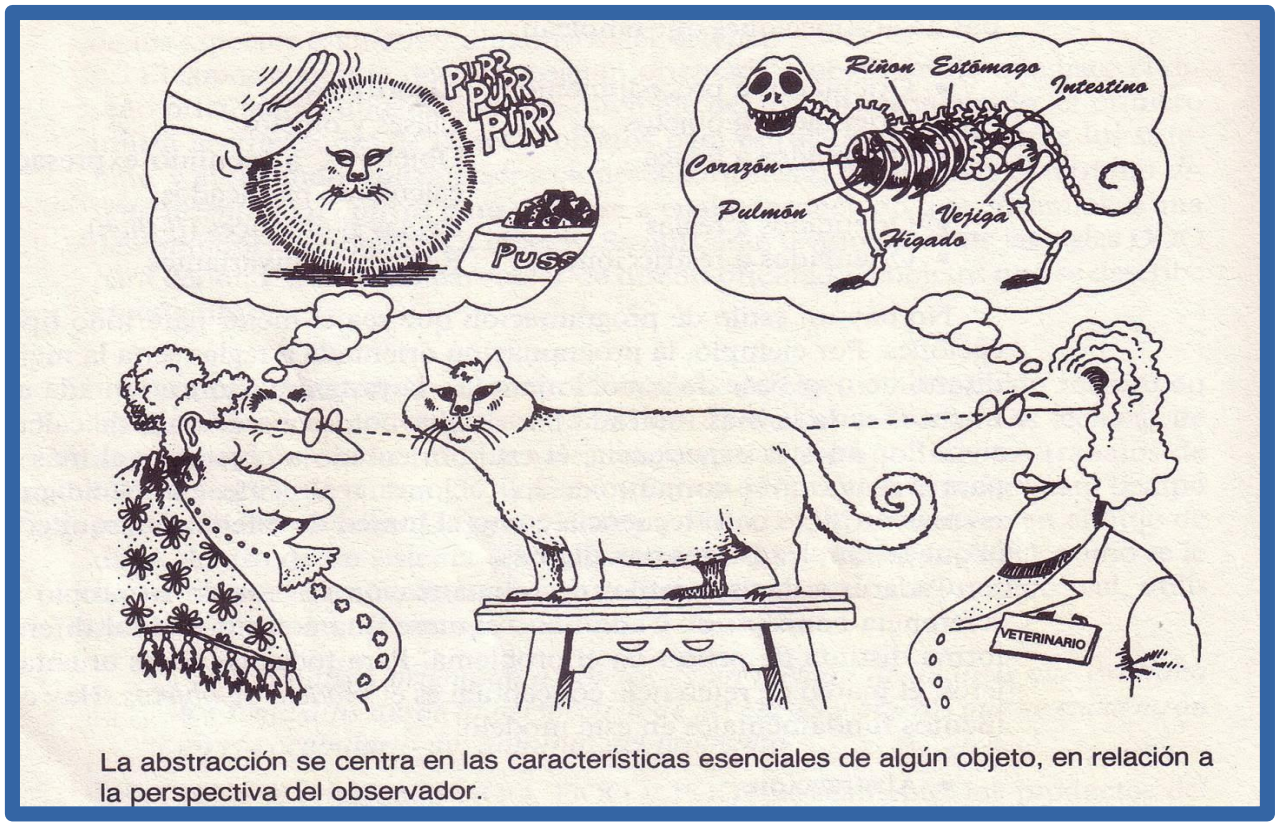
argumentos

```
Buen día María Jeréz
```

salida

Abstracción de la realidad al modelar con objetos

La Abstracción es una descripción especial simplificada de un sistema que hace énfasis en ciertos rasgos y suprime otros.



La abstracción se centra en las características esenciales de algún objeto, en relación a la perspectiva del observador.

Por ejemplo, la perspectiva de ver a un gato es muy distinta entre una abuela amorosa y un médico veterinario. La abuela hará una abstracción fijándose en rasgos afectivos y de cuidado mientras que el veterinario lo verá como un objeto anatómico-fisiológico de estudio.

Abstracción de la realidad al modelar con objetos

¿De qué hablamos cuando hablamos de abstracción?

- Es el proceso de supresión de detalles respecto de un fenómeno, entidad o concepto.
- El objetivo es concentrarse en los aspectos más significativos.

Algunas definiciones

- Un concepto general formado a partir de la extracción de características comunes tomadas de ejemplos específicos.
- El proceso mental donde las ideas son separadas de los objetos concretos.

¿Qué es el diseño orientado a objetos?

- Un diseño Orientado a Objetos expresa, sin necesidad de escribir código, como colaboran los objetos para realizar sus actividades.
- Han existido diversos métodos y técnicas orientadas a objetos, con muchos aspectos en común pero utilizando distintas notaciones.

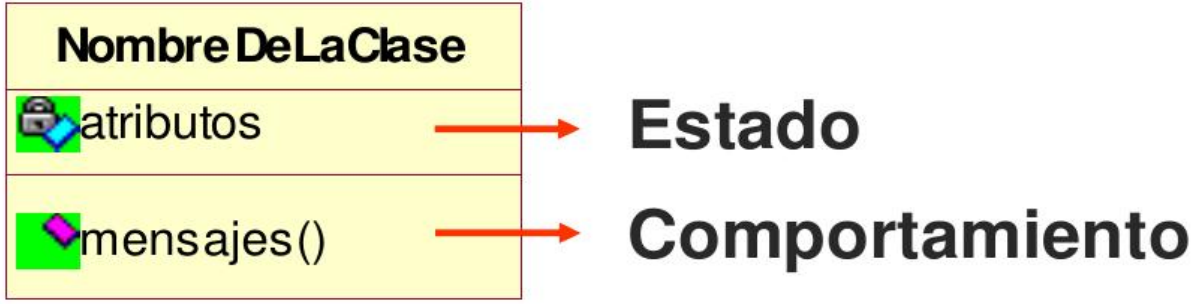
- Como resultado de varias de ellas, surgió el Lenguaje Unificado de Modelado (del Unified Modeling Language – UML) como estándar para el modelado de sistemas de software. En 1994, Booch, Rumbaugh (OMT) y Jacobson (Objectory) deciden unificar sus métodos y nace el UML.

“Es un lenguaje de propósito general para el modelado orientado a objetos”

Notación UML. Diagramas de Clase.

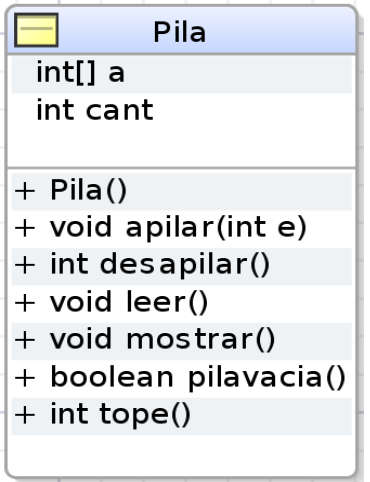
- Posee distintos tipos de diagramas, pero ahora sólo el Diagrama de Clases.
- En el diagrama de clases, un clase se describe en función de sus atributos y mensajes.

La notación para una clase es la siguiente:



Ejemplo

```java
public class Pila
{
    int[] a = new int[50];
    int cant;

    public Pila()
    {
        super();
        this.cant = 0;
    }

    public void apilar(int e)
    {
        this.a[cant] = e;
        this.cant++;
    }

    public int tope()
    {
        return (a[cant - 1]);
    }

    public int desapilar()
    {
        this.cant--;
        return (a[cant]);
    }

    public boolean pilavacia()
    {
        return (cant == 0);
    }

    public void leer()
    {
        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        this.apilar(i);
    }

    public void mostrar()
    {
        int i = 0;
        while (i < this.cant)
        {
            System.out.println(this.a[i]+ "    ");
            i++;
        }
    }
}
```



Ejemplo

```
public static void main(String args[])
{
    Pila p1 = new Pila();
    Pila p2 = new Pila();
    int i = 0;
    while (i < 4)
    {
        p1.leer();
    }
    p1.mostrar();
    p2.apilar(p1.desapilar());
    p2.apilar(p1.desapilar());
    p2.mostrar();
}
```