

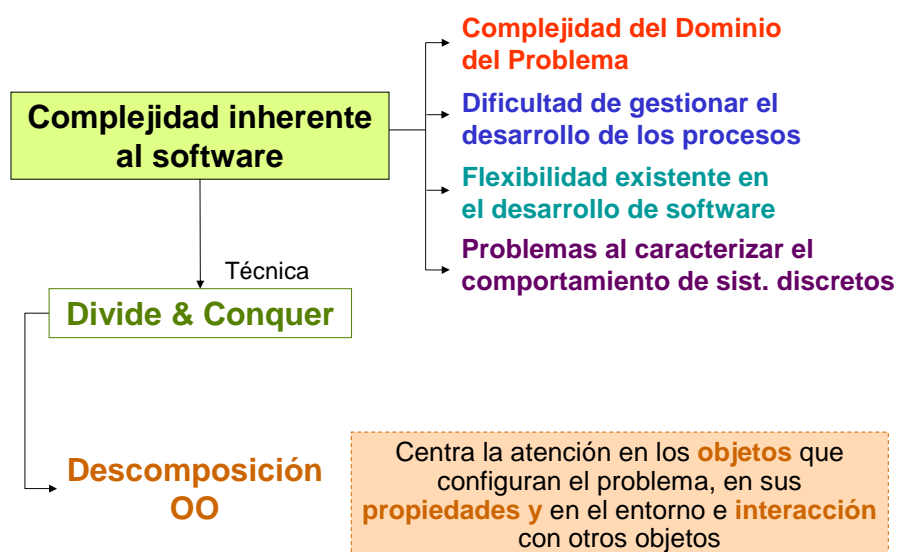


## UNIDAD II

### PARADIGMAS DE PROGRAMACIÓN

Programación Orientada a Objetos  
Ideas Fundamentales

## Desarrollo de Software

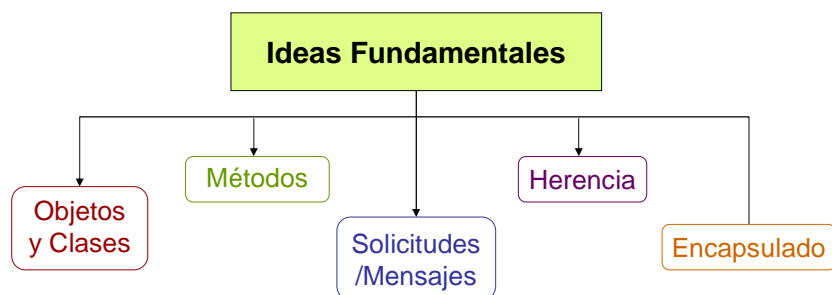


# Programación Orientada a Objetos (POO)

Método de implementación en el que los programas son organizados como **colecciones cooperativas de objetos**, cada uno de los cuales representa una **instancia de alguna clase**, y cuyas clases son miembros de **jerarquías de clases** unidas a través de una relación de herencia

3

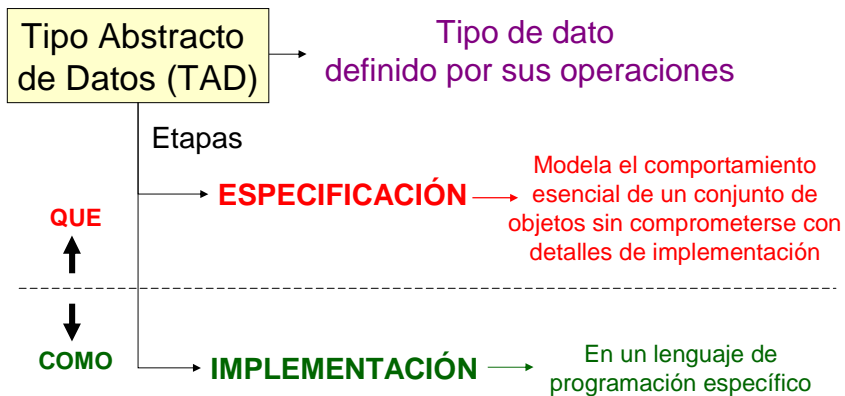
## Orientación a Objetos



4

# Clases y Tipos Abstractos de Datos

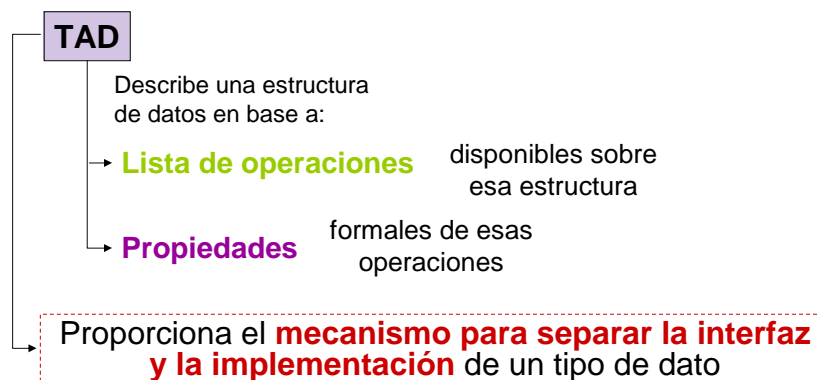
Objetos  
y Clases



5

# Clases y Tipos Abstractos de Datos

Objetos  
y Clases



6

# Ejemplo - TAD Pila

Objetos  
y Clases

## Types

Pila[X]

TAD Genérico. La **genericidad** permite que una misma especificación sirva para distintos propósitos

## Functions

crear:  $\rightarrow$  Pila[X]  
es-vacía: Pila  $\rightarrow$  boolean  
insertar:  $X * \text{Pila[X]} \rightarrow \text{Pila[X]}$   
eliminar:  $\text{Pila[X]} \rightarrow \text{Pila[X]}$   
cabeza:  $\text{Pila[X]} \rightarrow X$

No es un servicio que proporcionan los objetos del TAD, sino un servicio que proporciona el TAD para crear objetos

## Preconditions

pre eliminar (s : Pila[X]) = (not es-vacía(s))  
pre cabeza (s : Pila[X]) = (not es-vacía(s))

A este conjunto de servicios se le llama su **interfaz o protocolo**.

## Axioms

for all x: X, s: Pila[X]:  
es-vacía(crear());  
not es-vacía(insertar(x,s));  
cabeza(insertar(x,s)) = x;  
eliminar(insertar(x,s)) = s

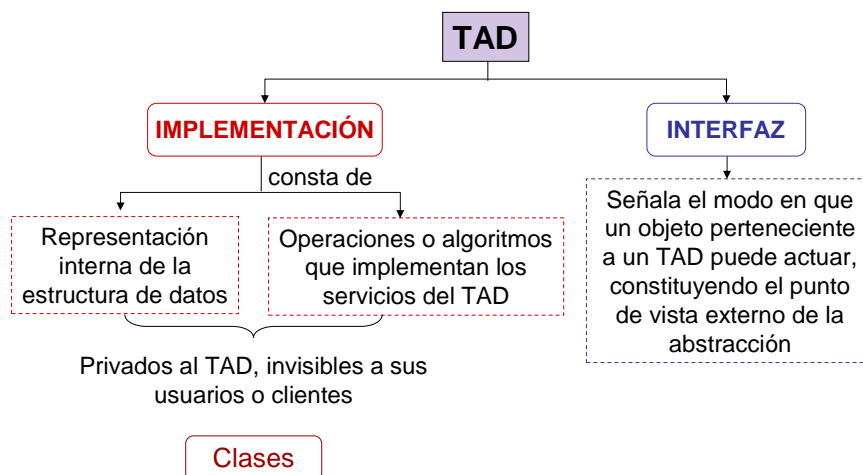
La especificación de un TAD  
captura las propiedades  
comunes mediante un tipo  
parametrizado

End

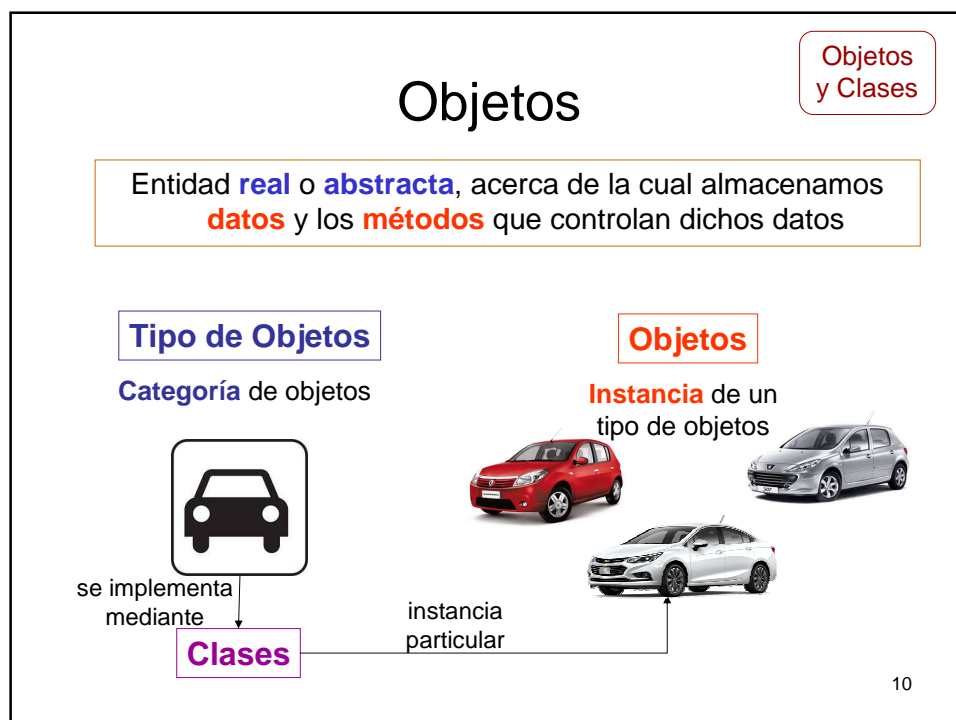
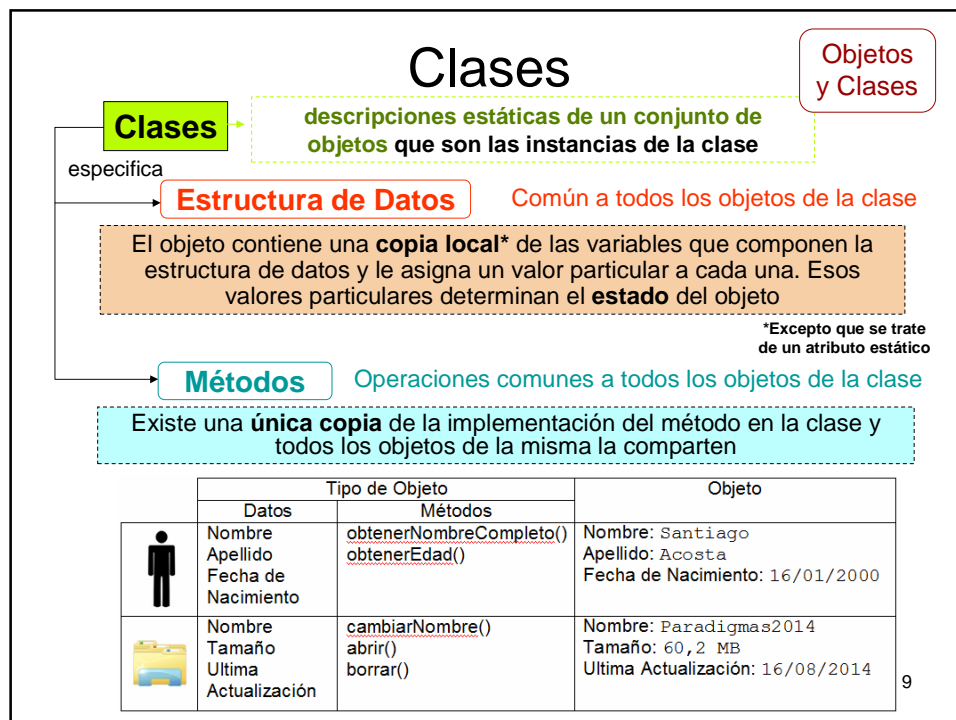
7

# Clases y Tipos Abstractos de datos

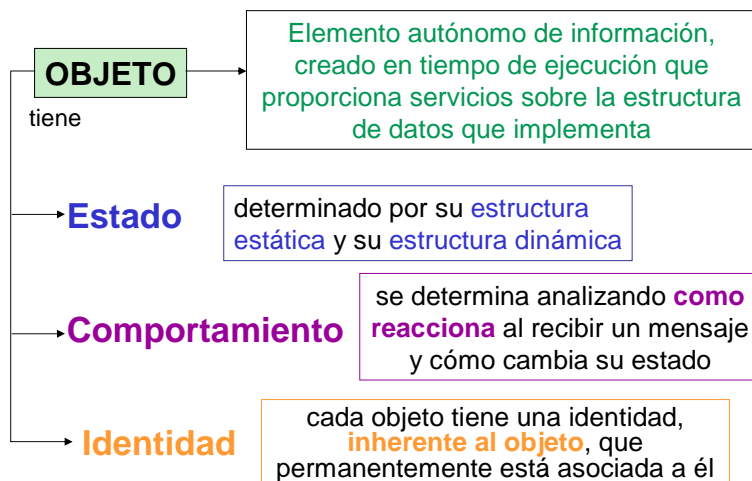
Objetos  
y Clases



8

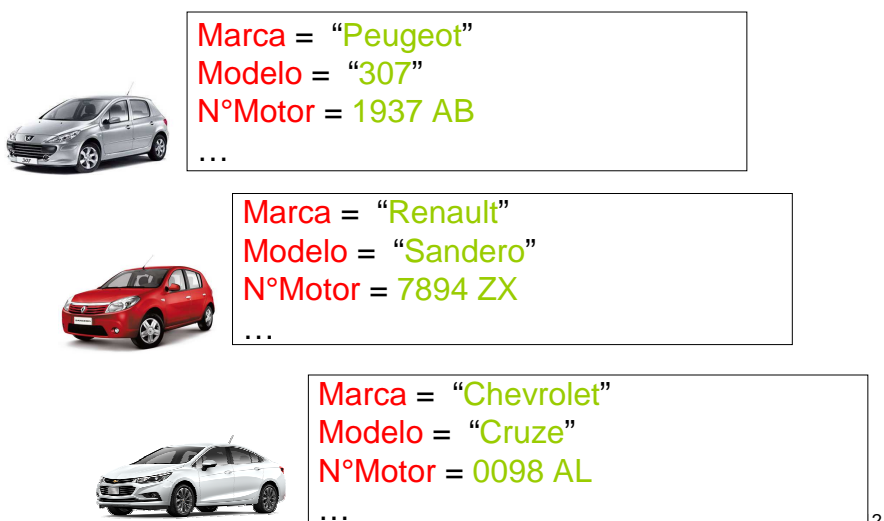


# Objetos



11

# Estado de un Objeto



2

# Identidad de un Objeto

Objetos  
y Clases

## Técnicas

Utilizadas para identificar un objeto

Utilizar direcciones o referencias de memoria

Utilizar nombres definidos por el usuario

Mediante claves identificadoras

En el caso de  
colecciones

13

# Identidad de un Objeto

Objetos  
y Clases

- En los primeros lenguajes de programación, la **identidad del objeto se correspondía con su nombre**, siendo imposible separar ambos conceptos.
- Cuando se incorpora a los lenguajes de programación la **asignación dinámica** de memoria **desaparece esta correspondencia**.
- El concepto de **referencia**, que permite que un mismo objeto tenga varios nombres ("**alias**"), **desvirtúa la relación nombre/identidad** del objeto.
- Otro concepto que muestra esta **disociación entre el nombre del objeto y su identidad**, se produce con el uso de **subprogramas**

14

# Identidad de un Objeto

Objetos  
y Clases

## Claves Identificadoras

Este mecanismo aplica como identificación de un objeto, un **conjunto de atributos** del mismo que son únicos y lo diferencian de los demás

Confunde el concepto de identidad con el de valor de ese o esos atributos

## Problemas

Modificación de  
claves identificadoras



No uniformidad

Uniones no  
naturales

# Métodos

Métodos

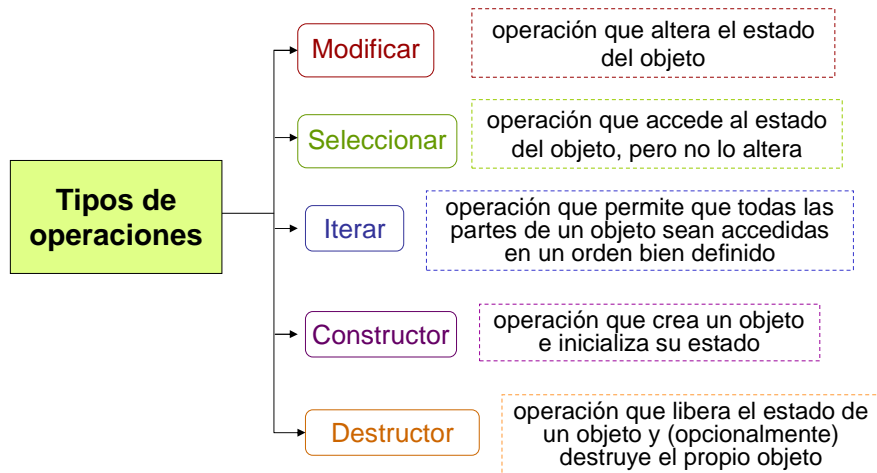
Especifican la forma en que **controlan los datos de un objeto** y **representan el comportamiento** de ese tipo de objetos

	Tipo de Objeto		Objeto
	Datos	Métodos	
	Nombre Apellido Fecha de Nacimiento	<u>obtenerNombreCompleto()</u> <u>obtenerEdad()</u>	Nombre: Santiago Apellido: Acosta Fecha de Nacimiento: 16/01/2000
	Nombre Tamaño Última Actualización	<u>cambiarNombre()</u> <u>abrir()</u> <u>borrar()</u>	Nombre: Paradigmas2014 Tamaño: 60,2 MB Última Actualización: 16/08/2014



# Comportamiento de un Objeto

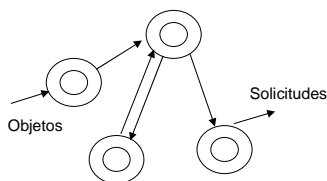
Métodos



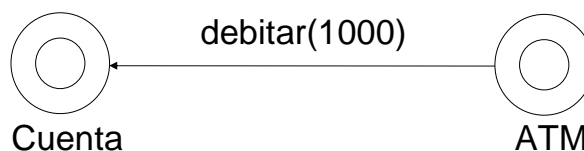
17

# Mensajes

Solicitudes /Mensajes

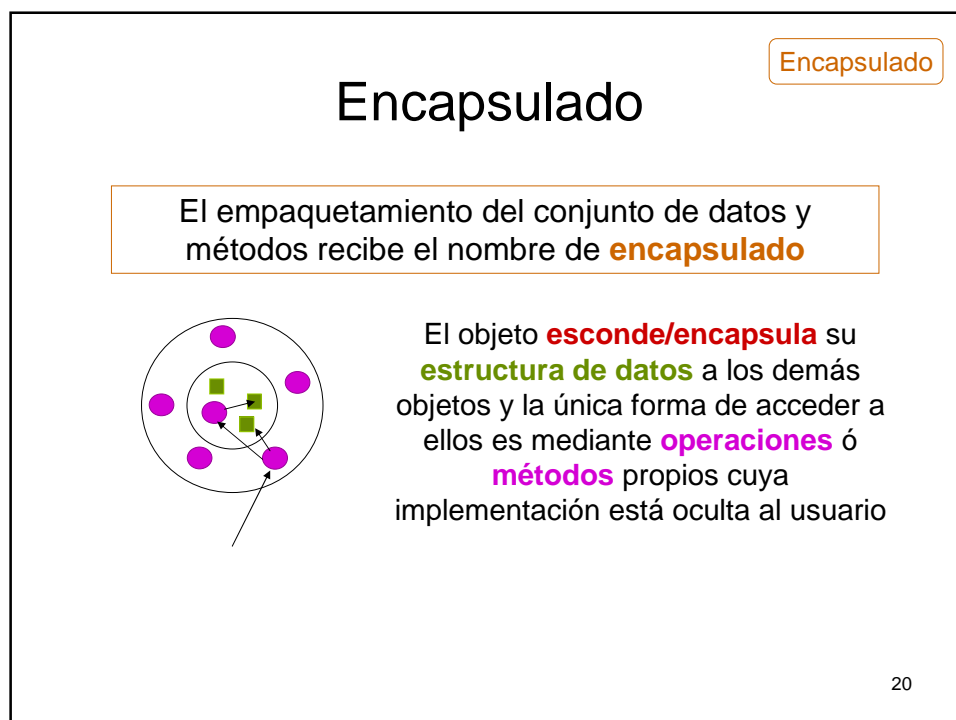
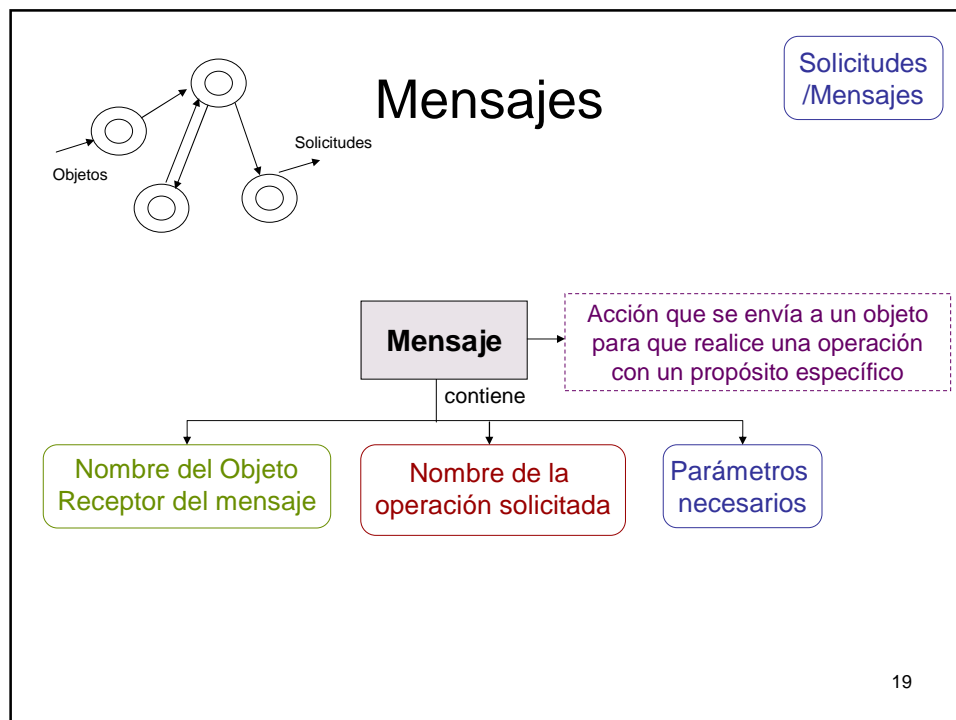


Los objetos se comunican mediante solicitudes. Una solicitud es un **mensaje** que **especifica la realización de una operación** con uno o más objetos como parámetros que, de manera opcional, devuelve un resultado



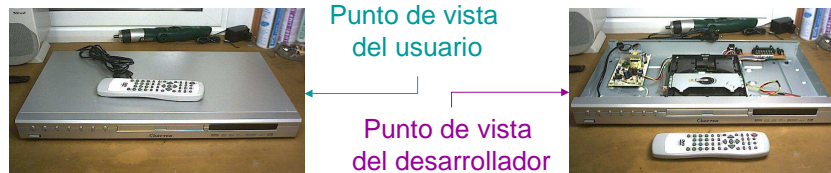
Los mensajes se usan para especificar una comunicación entre objetos

18



# Encapsulado

El encapsulado implica el **ocultamiento de los detalles de implementación** de un objeto respecto de su usuario



## Encapsulado

### Ventajas

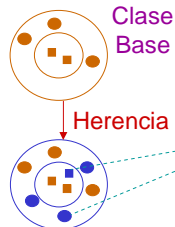
- **Evita la interferencia** con los aspectos internos
- **Oculto la complejidad** de sus componentes
- **Evita corrupción de los datos**
- **Protege los datos** de un uso arbitrario o no pretendido

21

# Herencia

Un tipo de objetos de alto nivel puede **especializarse** en sub-tipos ó tipos de objetos de niveles más bajos

**Hereda** propiedades de su clase padre o predecesor



Una clase puede tener sus propios **métodos** y su propia **estructura de datos**, además de los que herede de su superclase.

## Herencia múltiple

Cuando una clase hereda propiedades de más de una superclase

22

## Beneficios de la tecnología OO

- **Reutilización:** para maximizar la reutilización se construyen las clases de modo que se puedan adaptar.
- **Estabilidad:** las clases diseñadas para la reutilización repetida se vuelven estables.
- **El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel:** el encapsulado oculta los detalles de implementación de las clases y hace que clases complejas sean fáciles de utilizar.
- **Creciente biblioteca de tipos de objetos**
- **Se construyen clases cada vez más complejas:** El software se crea a partir de clases ya existentes y probadas. Esto permite construir componentes complejos de software.

23

## Beneficios de la tecnología OO

- **Diseño de mayor calidad:** Ya que se integran a partir de componentes ya probados que han sido verificados y pulidos varias veces.
- **Diseño más rápido**
- **Confiabilidad:** El software construido a partir de clases estables ya probadas tiene menos fallas que el software elaborado a partir de cero.
- **Integridad:** Las estructuras de datos sólo pueden ser utilizadas por métodos específicos.
- **Programación más sencilla:** Los programas se conforman a partir de pequeñas piezas, cada una de las cuales se crea fácilmente.

24

## Beneficios de la tecnología OO

- **Mantenimiento más sencillo:** El mantenimiento de Sistemas OO es mucho más sencillo que el mantenimiento de sistemas convencionales.
  - Los eventos **cambian el estado** de los objetos. La mayoría de estos cambios de estados requiere pequeñas partes de código menos propensas a errores.
  - Cada objeto lleva a cabo una función específica e independiente de los demás objetos
- **Modelado más realista:** El análisis OO modela el área de aplicación de manera que sea lo más cercana posible a la realidad.
- **Mejor comunicación entre los profesionales del sistema y los empresarios:** Los empresarios comprenden más fácilmente el paradigma OO.
- Etc.

25