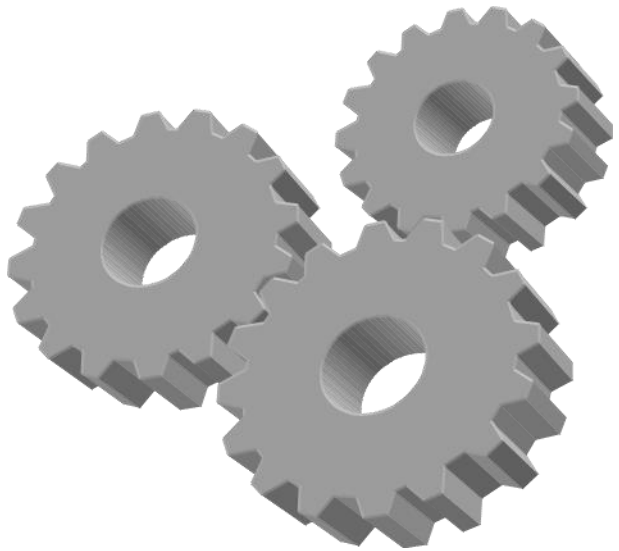
Introducción al Paradigma de Objetos



Mapa del Curso

Los Paradigmas

Modelado de Objetos

Introducción al Paradigma de Objetos

- Clases y Objetos
- Atributos, Tipos, Operaciones, Encapsulamiento
- Constructores
- Herencia

Introducción a UML

Proyecto Integrador

Los Paradigmas

Un **Paradigma** es un modelo o patrón en cualquier disciplina científica

Un **Paradigma de Programación** es una propuesta tecnológica que es adoptada por una comunidad de programadores cuyo núcleo central es incuestionable en cuanto a que unívocamente trata de resolver uno o varios problemas claramente delimitados.

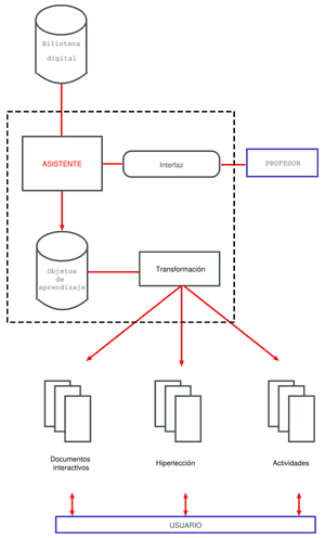
El **Paradigma de Programación Orientada a Objetos** es la implementación de un Paradigma de Programación

Que es un Modelo ?

Un modelo es una **abstracción** de la realidad

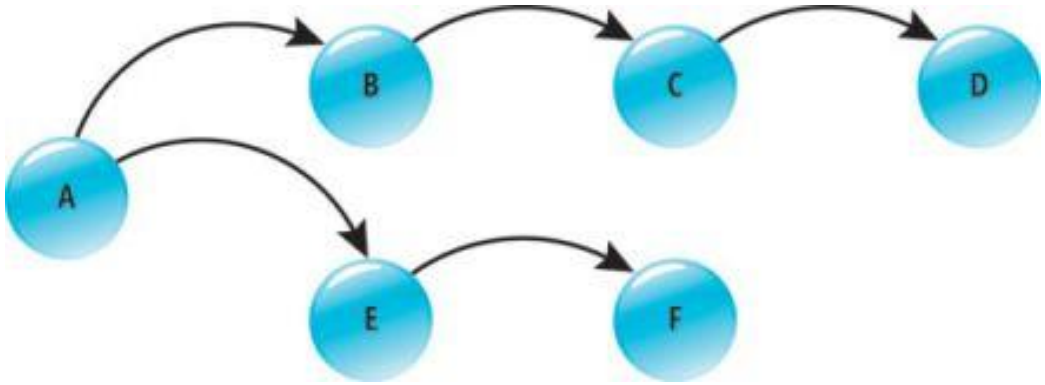
Es una **simplificación de la realidad**, con el objetivo final de pasar del modelo a producto real.

Los Sistemas de Información deben ser modelados previo a ser construidos



El Modelado Orientado a Objetos

Consiste en interpretar un sistema como partes independientes que se comunican entre sí



Las partes independientes se denominan **Objetos**

La comunicación entre los objetos se realiza a través de **Mensajes**

El Software Orientado a Objetos

- ✓ Altamente Escalable
- ✓ Fácil de Mantener
- ✓ Fácil de Reutilizar
- ✓ Muy Simple

Que es una Clase

Es una plantilla, es un molde que permite construir objetos

Representa ideas del mundo real, en forma genérica

Dentro de un sistema, las clases suelen detectarse como **sustantivos en singular**

Poseen atributos y métodos

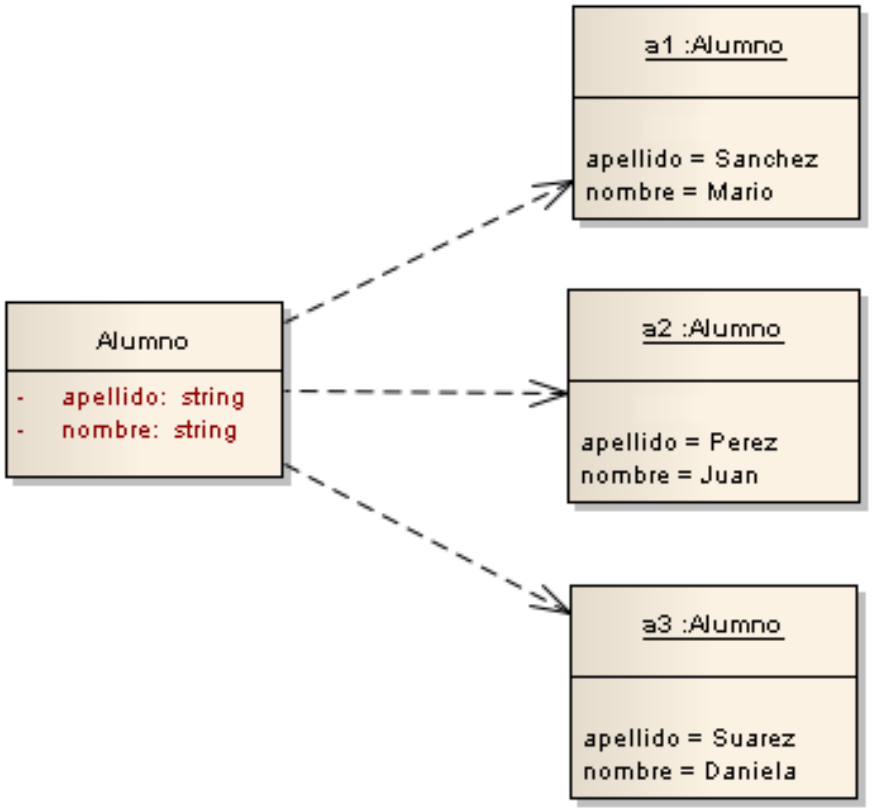
Ejemplos de clases: Auto, Empleado, CajaDeAhorro, Alumno

Que es un Objeto

Un objeto es una la **instancia de una clase**, podemos decir que el objeto representa algo en particular

Poseen un **estado** (de acuerdo a sus atributos)

Poseen un **comportamiento** (realizan operaciones de acuerdo a sus métodos)

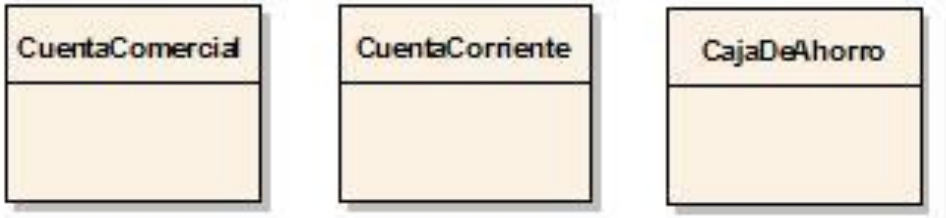


Ejercicio 1 – Detección de Clases

Identificar las clases iniciales y necesarias para construir un sistema de gestión bancaria

TIP: las clases se detectan como sustantivos en singular

Ejercicio #1 – Solución



Banco – Sucursal – GrupoFinanciero – Servicio

ClientePyme – ClienteCorporacion – ClienteIndividuo

CuentaComercial – CajaDeAhorro – CuentaCorriente

DirectorGeneral – DirectorRegional – DirectorDeSucursal

Ejercicio #1 – Codificación

```
class Banco  {

        // Atributos aquí


        // Métodos aquí

}
```

Que son los Atributos

Son **características** que posee una clase

Son variables contenidas y establecidas por los objetos, y normalmente cuentan con un tipo de dato asociado

Las atributos de una clase definen las características de sus objetos

Las clases definen los atributos, y los objetos “los completan”

Que es un Tipo de Dato

Es la forma de describir y/o almacenar un dato

Los tipos de datos numéricos mas conocidos son: **int, long, float, double**

Los tipos de datos de tipo caracter mas conocidos son: **String, char**

Para valores true/false el tipo de dato utilizado es **boolean** y para fechas se utiliza **Date**

Pueden ser otras clases!

Ejercicio #2 – Detección de Atributos

A partir de las clases detectadas anteriormente, identificar los atributos que considere necesarios en cada clase.

Cada atributo deberá tener establecido un tipo de dato. Deberá identificar al menos tres atributos por clase.

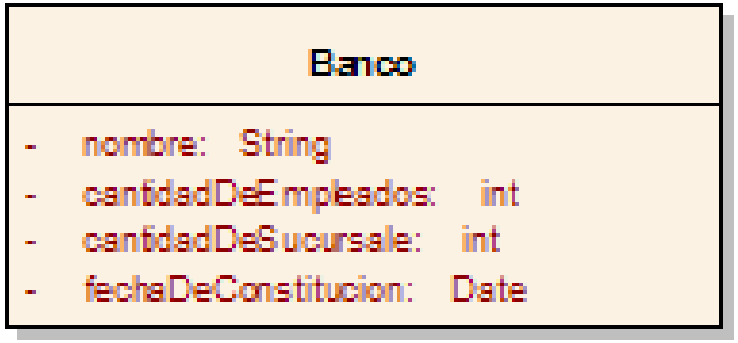
TIPS

- ✓ Los atributos son características de una clase
- ✓ Los tipos de datos a utilizar para este ejercicio son: String, int, long, float, double, Date

PREGUNTA

Donde ubicarías los atributos cantidadDeEmpleados, numeroDeCuenta y edad?

Ejercicio #2 – Solución



Sucursal: nombre, numero, direccion, cantidadDeEmpleados, cantidadDeClientes

ClientePyme: razonSocial, direccion, fechaDeAlta, **cuentaCorriente**

ClienteCorporacion: razonSocial, direccion, fechaDeAlta, cuentaCorriente

ClienteIndividuo: nombre, apellido, direccion, fechaDeAlta, cajaDeAhorro

Ejercicio #2 – Solución

Servicio: nombre, descripcion, fechaDeAlta
GrupoFinanciero: nombre, descripcion, fechaDeAlta

CajaDeAhorro: moneda, numero, cbu, saldo
CuentaCorriente: moneda, numero, cbu, saldo, **giroEnDescubierto**

DirectorGeneral: nombre, apellido, fechaDeNacimiento, fechaDeIngreso
DirectorRegional: nombre, apellido, fechaDeNacimiento, fechaDeIngreso
DirectorDeSucursal: nombre, apellido, fechaDeNacimiento, fechaDeIngreso

Ejercicio #2 – Codificación

```
class ClientePyme {

        // Atributos aquí
        String razonSocial;
        String direccion;
        Date fechaDeAlta;
        CuentaCorriente cuenta;

        // Métodos aquí

}
```

Que es una Operación

Las operaciones son acciones contenidas en una clase, y definen su comportamiento

Dentro de un sistema, las operaciones suelen detectarse como **verbos**

Desde la perspectiva de Diseño y Programación, se denominan **Métodos**

Desde la perspectiva de Análisis, se denominan **Operaciones**

Puede tener opcionalmente valores de entrada (Parámetros) y valores de salida (Valores de Retorno)

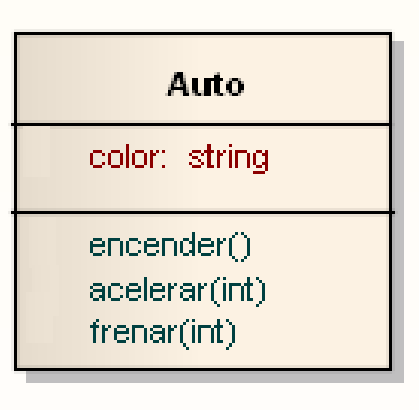
Procedimientos (no retornan un valor) vs. **Funciones** (retornan un valor)

Que es un Valor de Entrada o Parámetro

Los parámetros son valores enviados a una operación

La operación toma los parámetros como **valores de entrada**, y así puede realizar las acciones necesarias

Todos los parámetros deben tener un tipo de dato asociado



- Método encender() → sin parámetros

- Método acelerar(int) → recibe como parámetro la cantidad de “km” a acelerar

- Método frenar(int) → recibe como parámetro la cantidad de “km” que debe bajar de velocidad

Que es un Valor de Salida o Valor de Retorno

El **valor de salida** de una operación es un **valor retornado** por la operación luego de realizar cierto procesamiento

Los valores de entrada son **datos**, y los valores de salida son considerados **información**

Todos los valores de salida deben tener un tipo de dato asociado

Es posible retornar un **único valor de salida**

Ejercicio #3 – Detección de Operaciones

A partir de las clases y los atributos detectados en los ejercicios anteriores, identificar al menos dos operaciones que realiza cada clase

TIP: las operaciones son verbos

PREGUNTA: en que clase ubicarías las siguientes operaciones ?

- informarSaldo()
- informarSucursales()
- informarClientes()
- depositarDinero(monto),
- informarDatosDeCuenta(nroDeCuenta)

Ejercicio #3 – Solución

Banco:
informarCantidadDeEmpleados(), informarCantidadDeSucursales(), **informarSucursales(),** calcularFacturacionAnual(), calcularFacturacionMensual(), **informarClientes()**

Sucursal:
informarDireccion(), informarCantidadDeEmpleados, calcularFacturacionMensual(), calcularFacturacionAnual(), **informarClientes(), informarDatosDeCuenta(nroDeCuenta)**

ClientePyme: infomarMovimientosEnCuentas(), informarDatos()

Ejercicio #3 – Solución

ClienteCorporacion: infomarMovimientosEnCuentas(), informarDatos()

ClienteIndividuo: infomarMovimientosEnCuenta(), informarDatos()

CajaDeAhorro: **infomarSaldo(),** extraerDinero(monto), **depositarDinero(monto)**

CuentaCorriente: **infomarSaldo(),** extraerDinero(monto), **depositarDinero(monto)**

Servicio: informarNombre(), informarDescripcion()

Ejercicio #3 – Solución

GrupoFinanciero: informarFechaDeAlta(), informarDescripcion()

DirectorGeneral: informarDatos()

DirectorRegional: informarDatos()

DirectorDeSucursal: informarDatos()

Ejercicio #3 – Codificación

```
class CajaDeAhorro {

        // Atributos aquí
        float saldo;

        // Métodos aquí
        void informarSaldo() {
                // Imprime el atributo saldo
                print(saldo);
        }

        float obtenerSaldo(){
                // Retorna el saldo
                return saldo;
        }
}
```

Ejercicio #3 – Codificación

```
class CajaDeAhorro {

        // Atributos aquí
        float saldo;

        // Métodos aquí
        void depositarDinero(float unMonto) {
                // Actualiza el valor del atributo saldo
                saldo = saldo + unMonto;
        }

        void extraerDinero(float unMonto){
                // Actualiza el valor del atributo saldo, NO controla si monto > saldo
                saldo = saldo – unMonto;
        }
}
```