PROGRAMACION ORIENTADA A OBJETOS

Visual Basic .NET

CESAR DAVID FERNANDEZ GRUESO

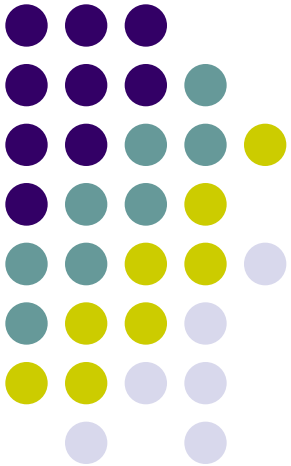
"Paradigma eficaz al servicio de la abstracción de problemas reales"

SENA Regional Cauca

CENTRO DE TELEINFORMATICA Y PRODUCCION INDUSTRIAL

TECNICO EN PROGRAMACION DE SOFTWARE

Vigencia 2009 - 2010



DEFINICION

- La programación orientada a objetos no debe confundirse con un lenguaje programación orientado a objetos.

- La POO es un paradigma, es otra forma de pensar , es una filosofía única a diferencia de un Lenguaje de Programación Orientado a Objetos el cual existen muchos y permiten hacer uso de ese paradigma con el animo de solucionar problemas reales mediante la abstracción de los diferentes agentes, entidades o elementos que actúan en el planteamiento de un problema.

- Ejemplo:

  a. **Problema:** Una persona necesita ver televisión.

  b. **Solución:** Existen 3 elementos o agentes que se pueden abstraer del problema:

ELEMENTO	DESCRIPCION
Persona	Tiene sus propios atributos: Color piel, Altura, genero, Color ojos, Cabello, etc. Y tiene un comportamiento: Ver , escuchar, hablar, etc.
Control Remoto	Tiene sus propios atributos: Tamaño, color, tipo, batería, etc. Y tiene un comportamiento: Enviar señal, codificar señal, cambiar canal, aumentar volumen, ingresar a menú, prender TV etc.
Televisor	Tiene sus propios atributos: pulgadas, tipo, numero parlantes, marca , etc. Y tiene un comportamiento: Decodificar señal, prender, apagar, emitir señal, emitir audio, etc.

DEFINICION

- En el problema planteado se especifican 3 elementos involucrados. Cada elemento posee sus propias características y sus propios comportamientos. En POO a estos elementos se les conoce bajo el nombre de **OBJETOS**.

- En POO a las características que identifican a cada objeto se le denominan **ATRIBUTOS** y a los comportamientos se les denominan **METODOS**.

ELEMENTO	DESCRIPCION
Persona	Tiene sus propios atributos: Color piel, Altura, genero, Color ojos, Cabello, etc. Y tiene un comportamiento: Ver , escuchar, hablar, etc.
Control Remoto	Tiene sus propios atributos: Tamaño, color, tipo, batería, etc. Y tiene un comportamiento: Enviar señal, codificar señal, cambiar canal, aumentar volumen, ingresar a menú, prender TV etc.
Televisor	Tiene sus propios atributos: pulgadas, tipo, numero parlantes, marca , etc. Y tiene un comportamiento: Decodificar señal, prender, apagar, emitir señal, emitir audio, etc.

DEFINICION DE CLASE

- Una **CLASE** es una plantilla mediante la cual se crean los diferentes objetos requeridos para la solución del problema. Los Objetos son instancias de las clases.
- Las clases son a los objetos como los tipos de datos son a las variables.
- Ejemplo: Se puede crear un objeto llamado Cesar. Este objeto es creado a partir de la clase Persona. Se puede crear otro objeto llamado: Patricia el cual pertenece a la clase Persona. Significa que a partir de la clase se pueden crear los objetos que se deseen.
- Ejemplo: Se puede crear un objeto llamado LCD LG, el cual pertenece a la clase Televisor.

ELEMENTO	DESCRIPCION
Persona	Tiene sus propios atributos: Color piel, Altura, genero, Color ojos, Cabello, etc. Y tiene un comportamiento: Ver , escuchar, hablar, etc.
Control Remoto	Tiene sus propios atributos: Tamaño, color, tipo, batería, etc. Y tiene un comportamiento: Enviar señal, codificar señal, cambiar canal, aumentar volumen, ingresar a menú, prender TV etc.
Televisor	Tiene sus propios atributos: pulgadas, tipo, numero parlantes, marca , etc. Y tiene un comportamiento: Decodificar señal, prender, apagar, emitir señal, emitir audio, etc.

DEFINICION DE OBJETO

- Es una instancia de una clase. Por lo tanto, los objetos hacen uso de los ***Atributos*** (variables) y ***Métodos*** (Funciones y Procedimientos) de su correspondiente Clase.

- Es una variable de tipo clase. *Por ejemplo*: El objeto Cesar es un objeto de tipo Clase: **Persona**.

- Permiten modelar entidades del mundo real. Por ejemplo: LCD LG pertenece a la clase Televisor. Resumiendo la clase televisor seria:

ATRIBUTOS
tipo. De tipo cadena. **Resolución.** De tipo cadena **Marca.** De tipo cadena.
METODOS
Emitir_Señal ( ) Emitir_Audio ( ) Decodificar_Señal (señal_entrada)

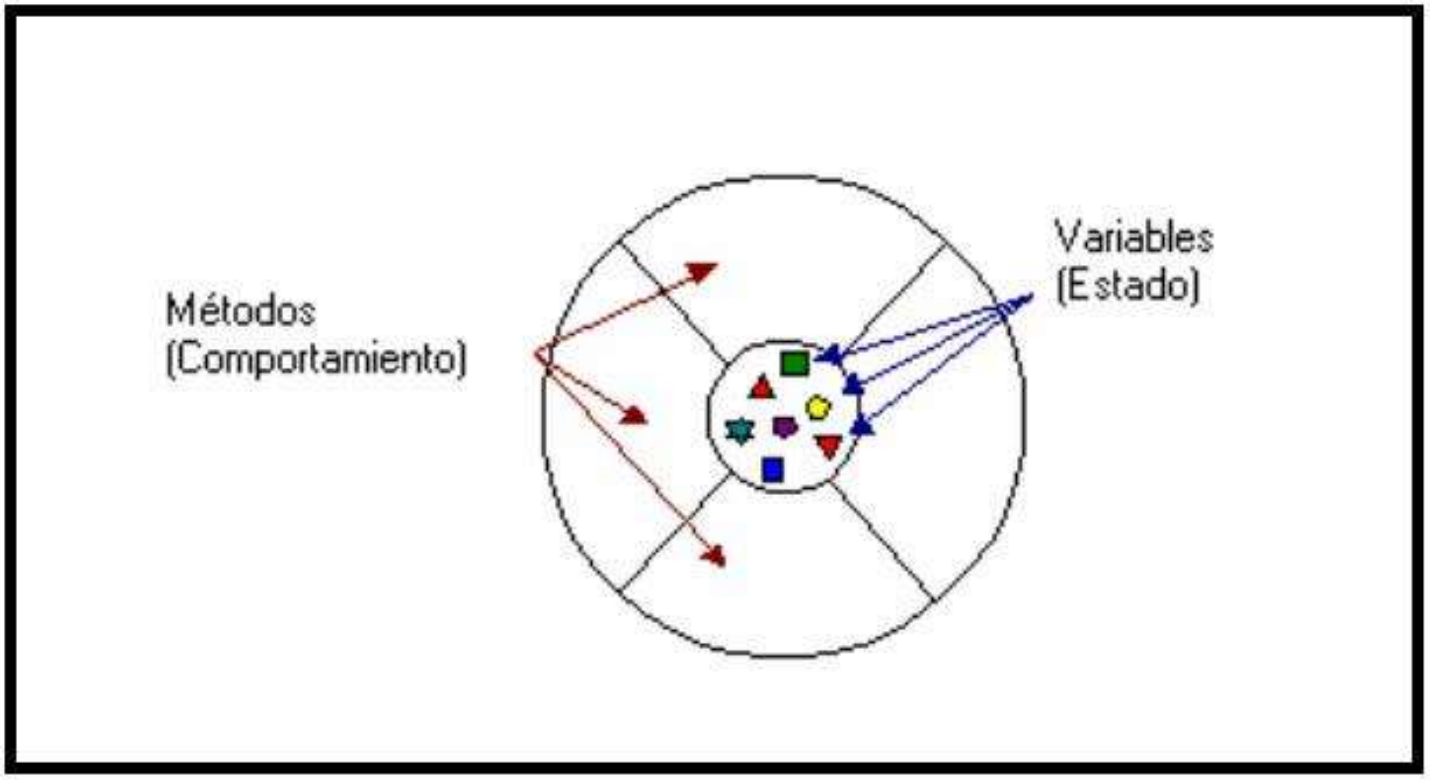
DEFINICION DE OBJETO

- Como se puede observar un objeto a través de su clase esta compuesto por 2 partes: Atributos o propiedades y Métodos que definen el comportamiento de dicho objetos a partir de sus atributos.

- Los atributos y los métodos pueden ser o no accedidos desde afuera dependiendo de la solución a plantear. Por lo general los atributos siempre se ocultan al exterior y algunos métodos quedan visibles al exterior para convertirse en la interfaz del objeto. **Encapsulamiento.**

ATRIBUTOS
tipo. De tipo cadena. **Resolución.** De tipo cadena **Marca.** De tipo cadena.
METODOS
Emitir_Señal ( ) Emitir_Audio ( ) Decodificar_Señal (señal_entrada)

DEFINICION DE OBJETO



IDENTIFICACION DE OBJETOS

- La primera tarea a la que se enfrenta un diseñador o programador en POO es la identificación e los objetos inmersos en el problema a solucionar.

- Los objetos generalmente se ubican en las siguientes categorías:

  - Cosas Tangibles: Avión, auto, producto, insumo.
  - Roles : gerente, cliente, vendedor, auxiliar, empleado.
  - Organizaciones o entidades: Empresa, colegio, proveedor, EPS.
  - Cosas intangibles: Vuelos, Servicios, Materias, programas.

EJEMPLO DE OBJETO

OTRO EJEMPLO:

- Se pretende modelar un objeto llamado CARRO el cual existe en el mundo real. Este objeto tiene unos atributos o variables: ***Vel_Max, Color, No_chasis, No_puertas, No._llantas***, ***tipo***. Unos comportamientos y métodos: ***Acelerar*** *(velocidad),* ***Frenar*** *(velocidad),* ***mover_cambio*** *(No_cambio),*

CLASE CARRO

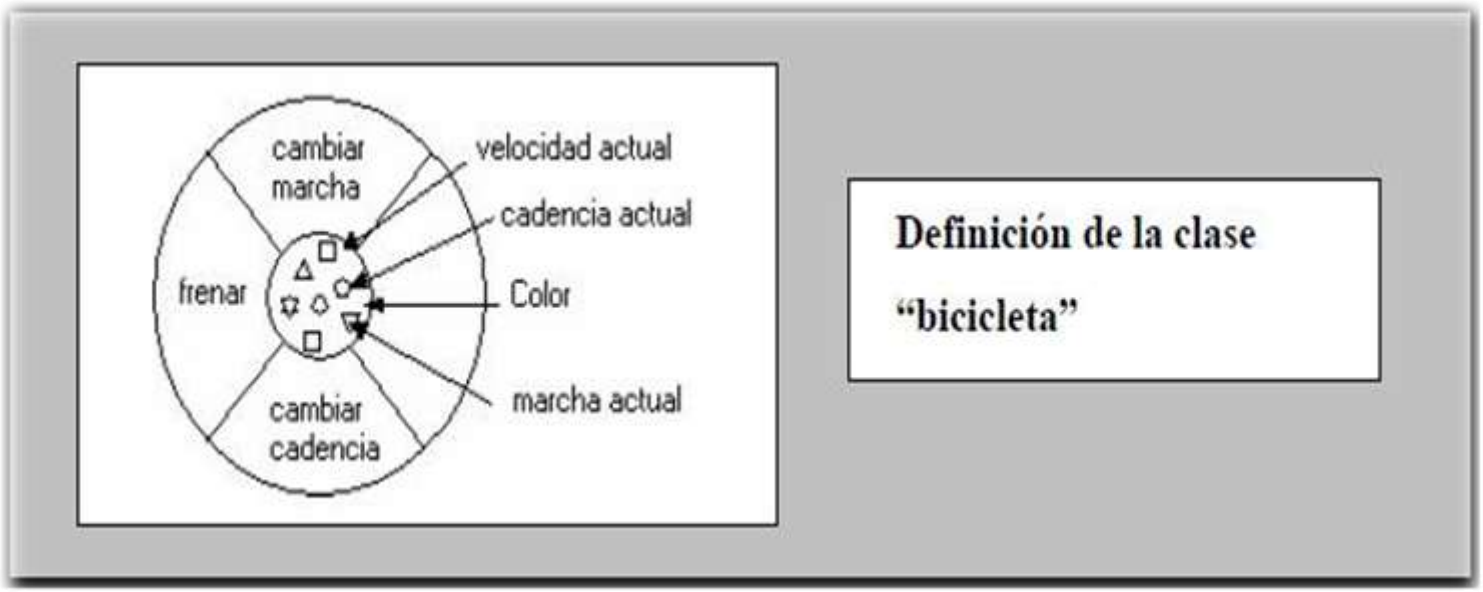
ATRIBUTOS

Vel_max. De tipo decimal.
Color. De tipo cadena
No_chasis. De tipo cadena.
No_puertas. De tipo entero.
No_llantas. De tipo entero.

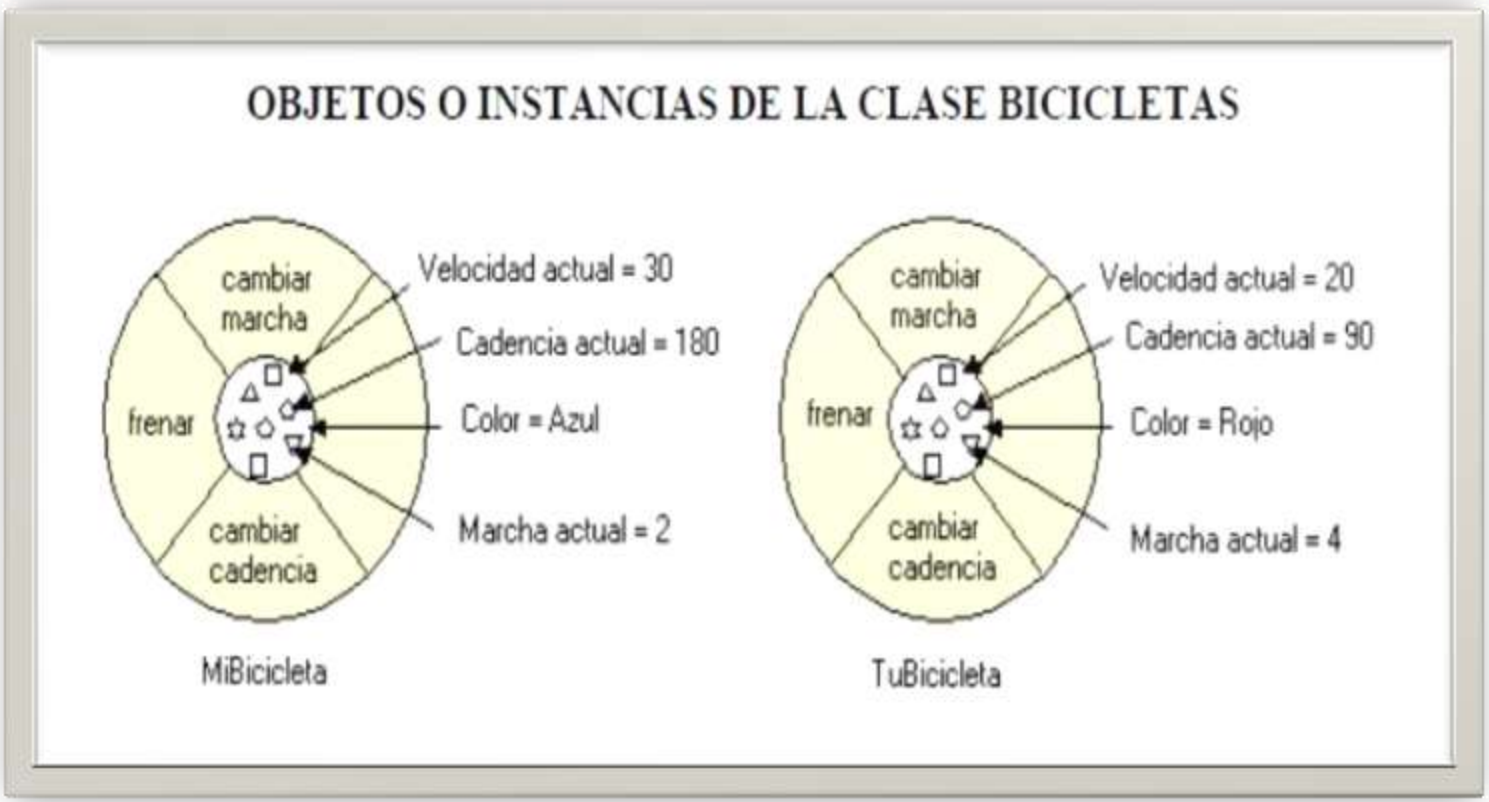
METODOS

Acelerar (Velocidad)
Frenar (Velocidad)
Mover_cambio (No_cambio)
Girar_derecha ( )
Girar_izquierda ( )

EJEMPLO CLASE



EJEMPLO OBJETOS



CARACTERISTICAS IMPORTANTES DE LA POO

- ABSTRACCION.
- ENCAPSULAMIENTO.
- MENSAJES.
- POLIMORFISMO.
- HERENCIA.

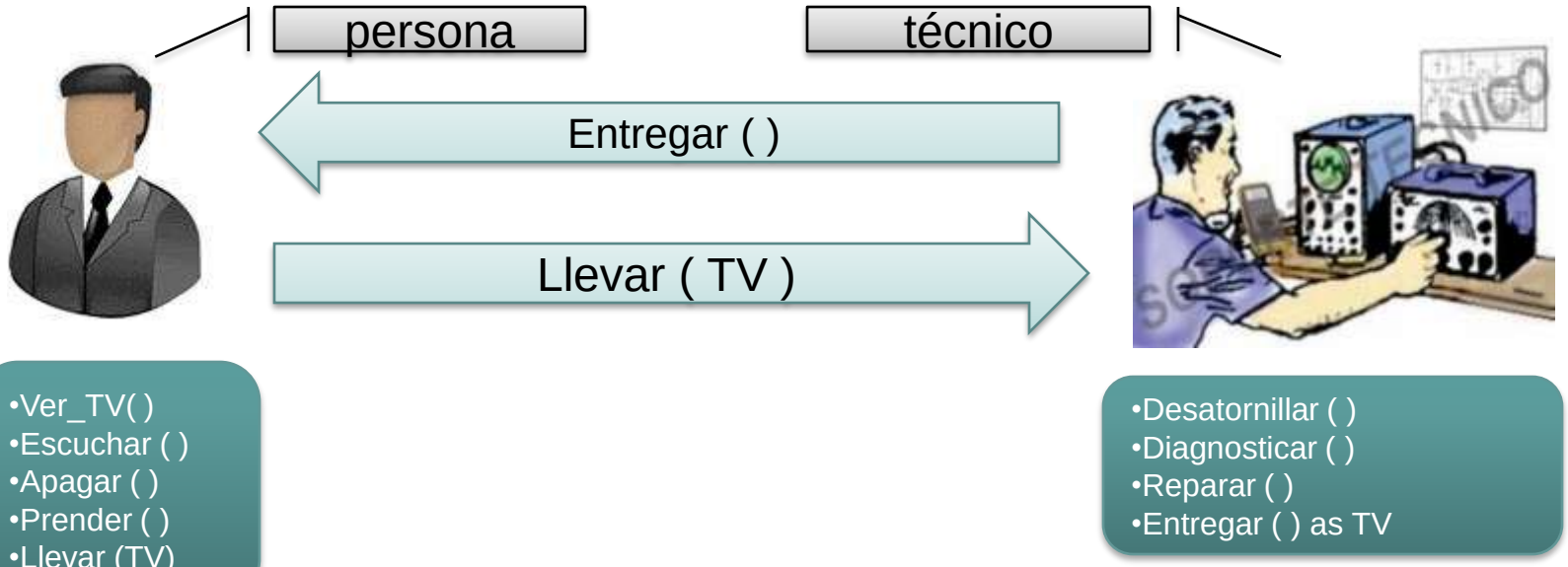
ABSTRACCION

- Es una de las principales características a tener en cuenta ya que permite vislumbrar los diferentes agentes u objetos implicados en un problema.
- Captar los atributos y métodos que conforman cada objeto y la relación que existen entre ellos.
- Resolver el problema en subproblemas donde cada objeto se haga cargo de cada subproblema.
- La comunicación entre objetos generan la solución general a todo el problema. (*Divide y vencerás*).

ENCAPSULAMIENTO

- Esta propiedad permite la ocultación de la información es decir permite asegurar que el contenido de un objeto se pueda ocultar del mundo exterior dejándose ver lo que cada objeto necesite hacer publico.

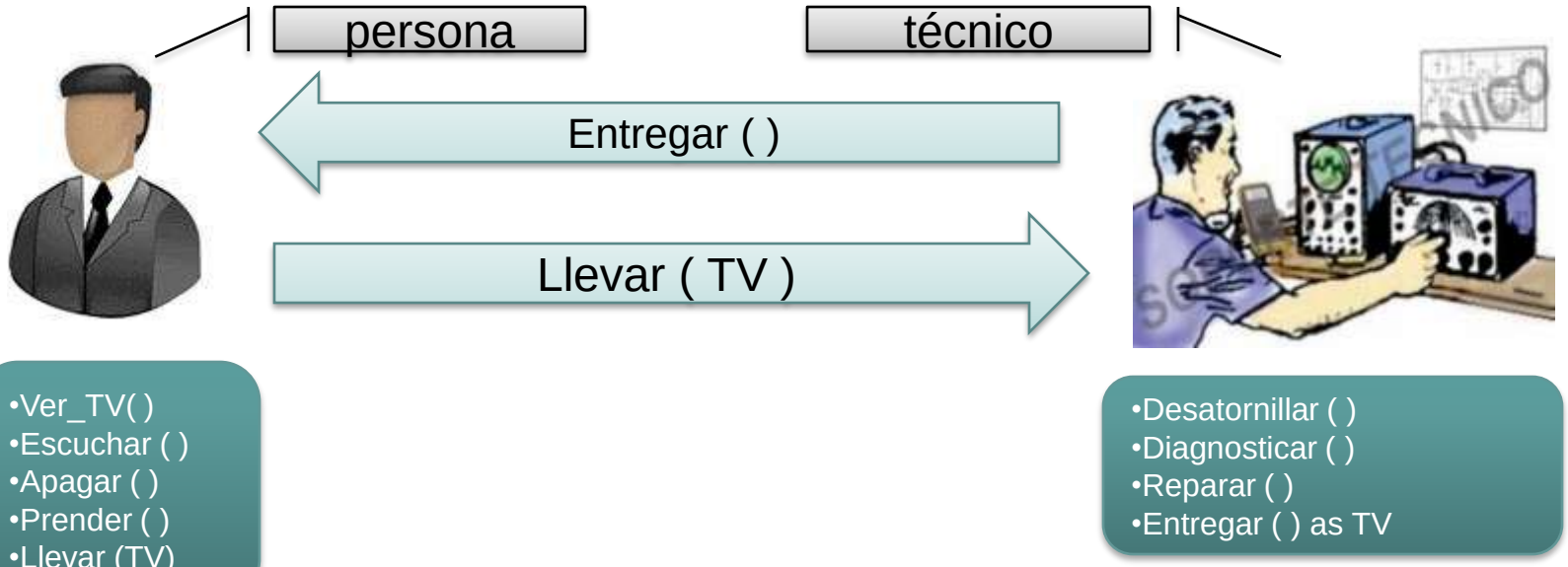
- **Ejemplo:** Una persona desea llevar su televisor descompuesto para que sea arreglado por un técnico.



ENCAPSULAMIENTO

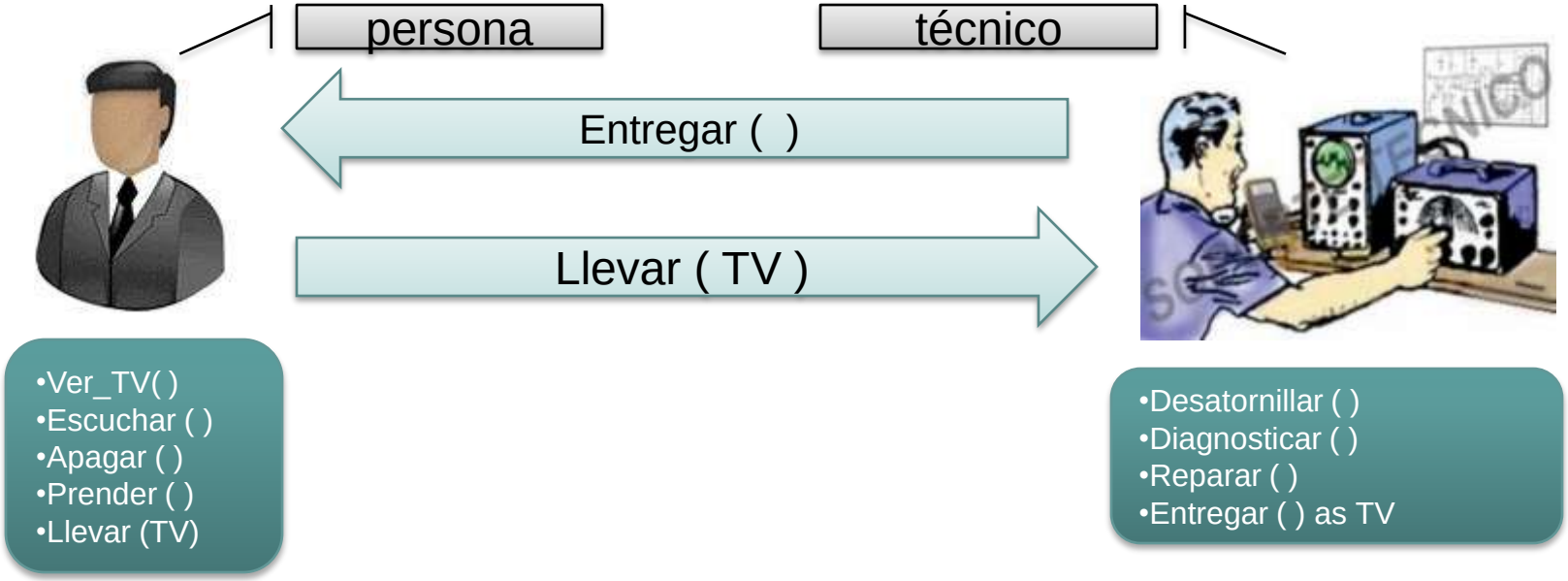
- Esta propiedad permite la ocultación de la información es decir permite asegurar que el contenido de un objeto se pueda ocultar del mundo exterior dejándose ver lo que cada objeto necesite hacer publico.

- **Ejemplo:** Una persona desea llevar su televisor descompuesto para que sea arreglado por un técnico.



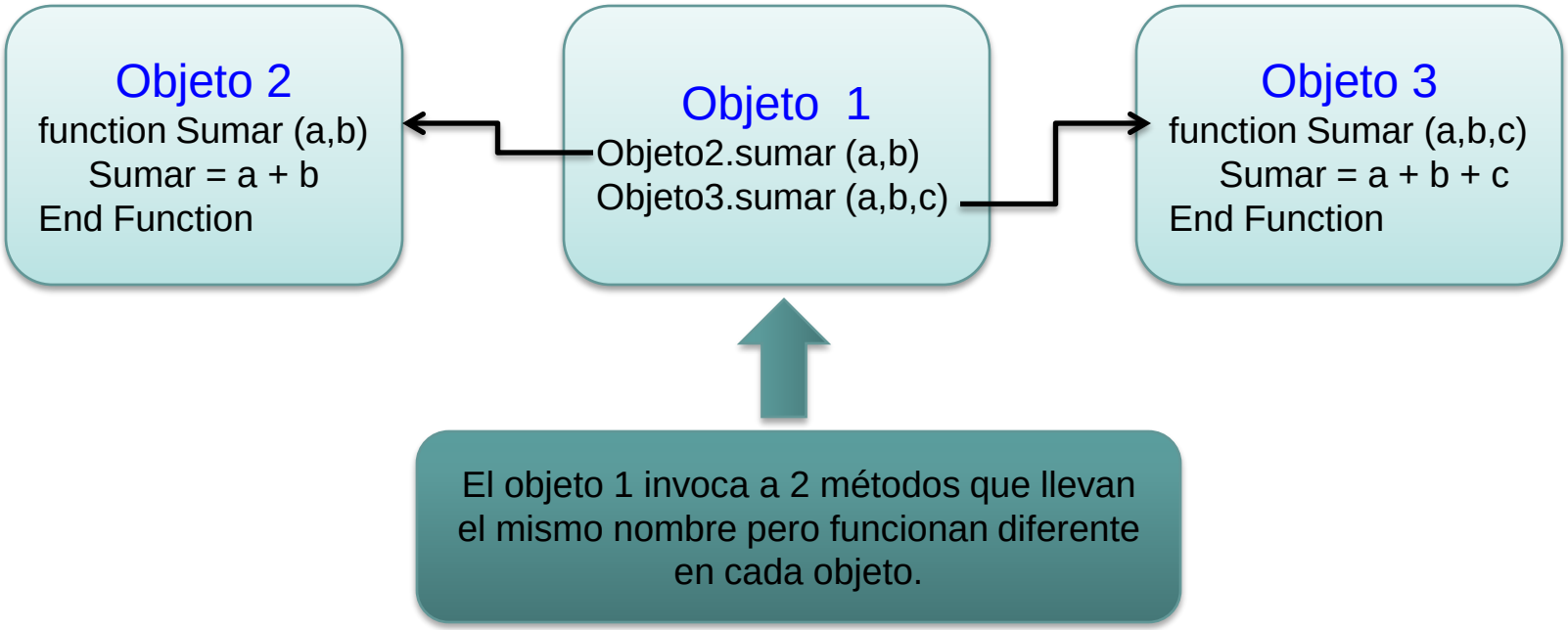
MENSAJES

- Un objeto sin comunicación con el mundo exterior no es de utilidad. La idea no es crear islas de objetos si no objetos relacionados.
- Los objetos interactúan entre ellos mediante ***mensajes.***
- ***Cuando un objeto A quiere que otro objeto B ejecute una de sus funciones o procedimientos miembro (Métodos de B), el objeto A manda un mensaje al objeto B.***



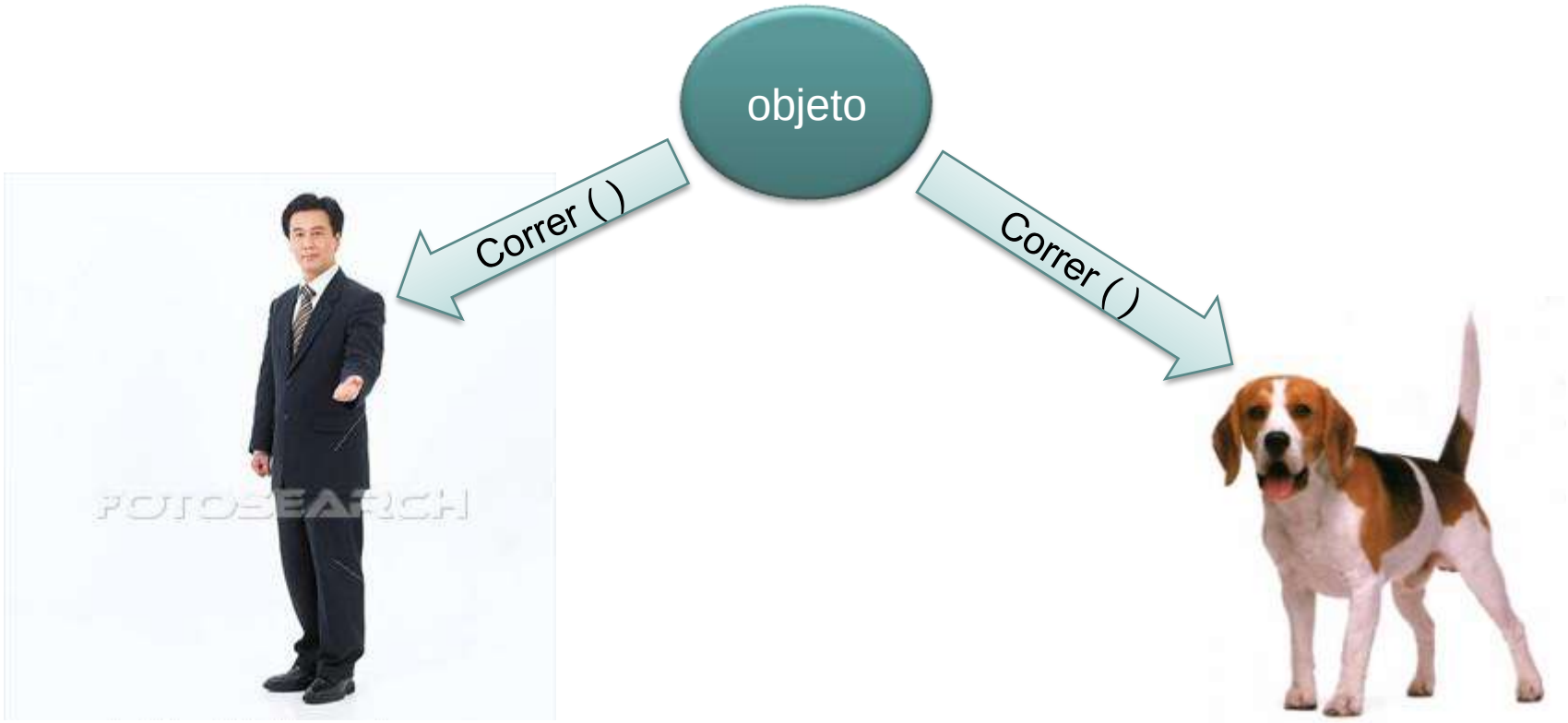
POLIMORFISMO

- Los comportamientos pueden ser identificados bajo el mismo nombre pero procesan información de manera diferente de acuerdo al objeto que lo contenga.



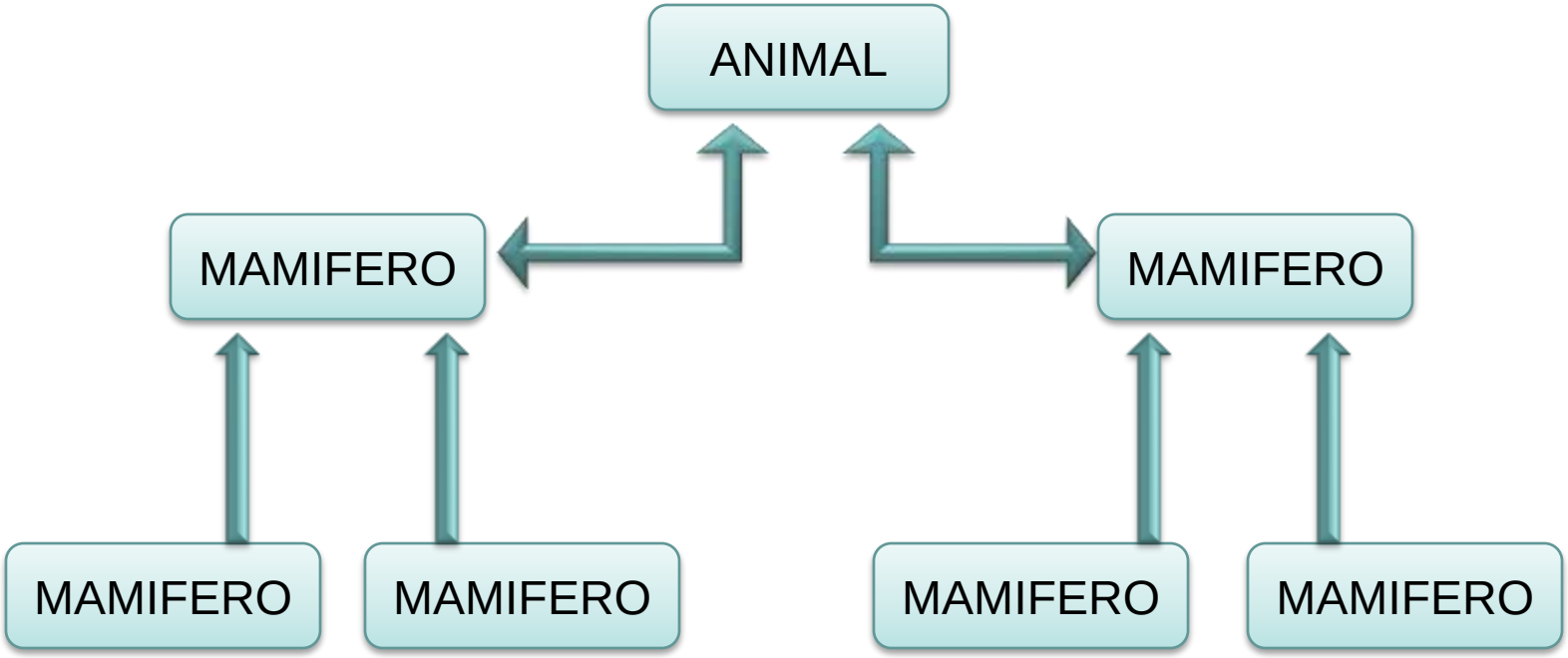
POLIMORFISMO

- Los comportamientos pueden ser identificados bajo el mismo nombre pero procesan información de manera diferente de acuerdo al objeto que lo contenga.



HERENCIA

- El mecanismo de herencia permite definir nuevas clases partiendo de otras ya existentes. Las clases que derivan de otras heredan automáticamente todo su comportamiento, pero además pueden introducir características particulares propias que las diferencian.



MODIFICADORES DE ACCESO

- Permiten controlar la forma de acceder a los atributos y métodos encapsulados dentro de una clase.

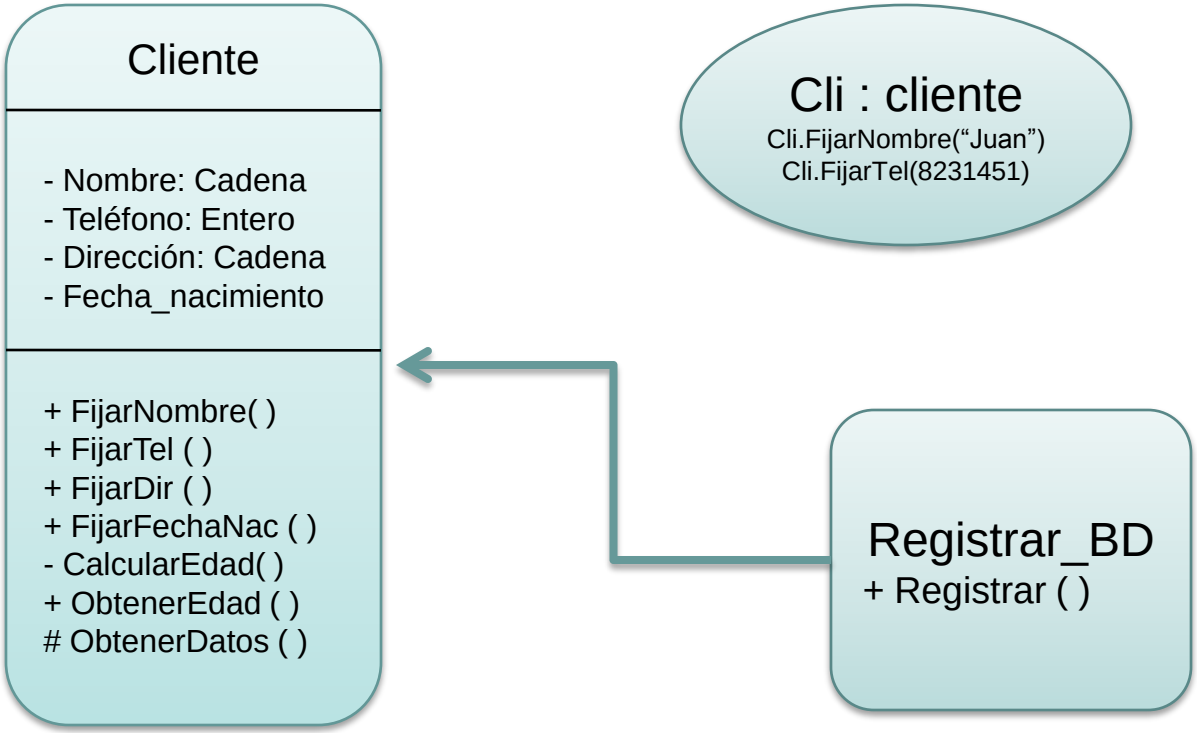
- **TIPOS**:

  - **PUBLICO:** Cualquier atributo o método Publico puede se accedido desde fuera de la clase. Se representa por (+).

  - **PRIVADO:** Cualquier atributo o método Privado NO puede se accedido desde fuera de la clase. Solo puede ser utilizado internamente en la clase. Se representa por (-).

  - **PROTEGIDO:** Cualquier atributo o método Protegido puede ser heredado por otra clase pero en esta ultima se convierten en elementos Privados. Se representa por (#).

MODIFICADORES DE ACCESO

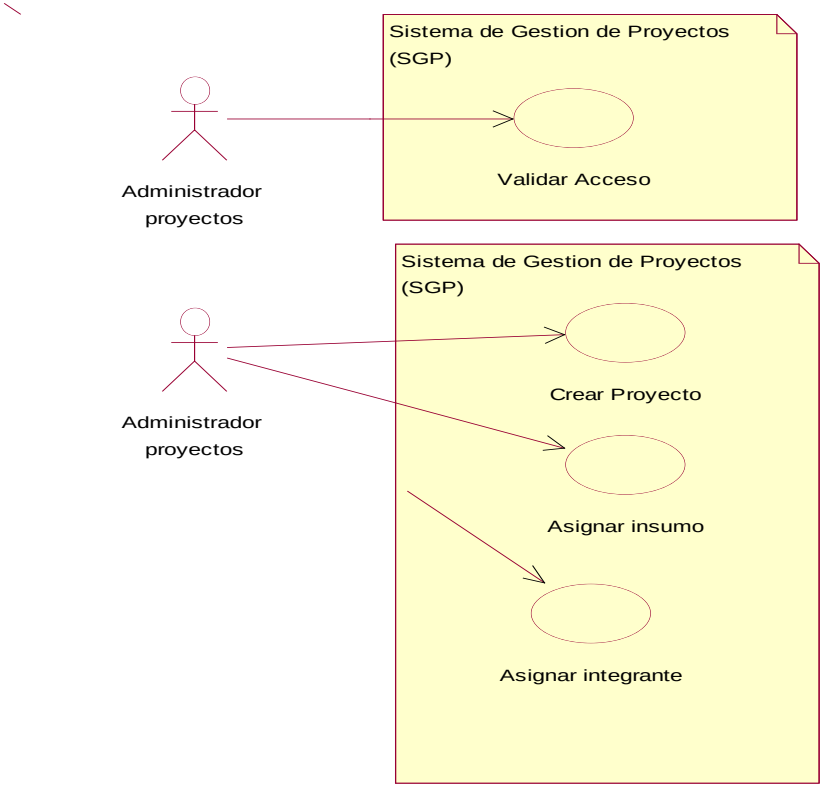


UML
(Unified Modeling Language)

- UML es un lenguaje de modelado para especificar el análisis y diseño de sistemas orientados a objetos.
- Permite diagramas los requerimientos funcionales del sistemas : ***Diagrama de Casos de Uso.***
- Permite abstraer mediante diagramas específicos las diferentes clases y objetos con sus respectivos atributos y métodos logrando especificar claramente las correspondientes relaciones o envió de mensajes entre objetos. ***Diagramas de Clases, Objetos y de Secuencia.***
- Permite visualizar los diferentes componentes donde se pondrá en funcionamiento el nuevo sistemas orientado a objetos. ***Diagramas de Despliegue.***

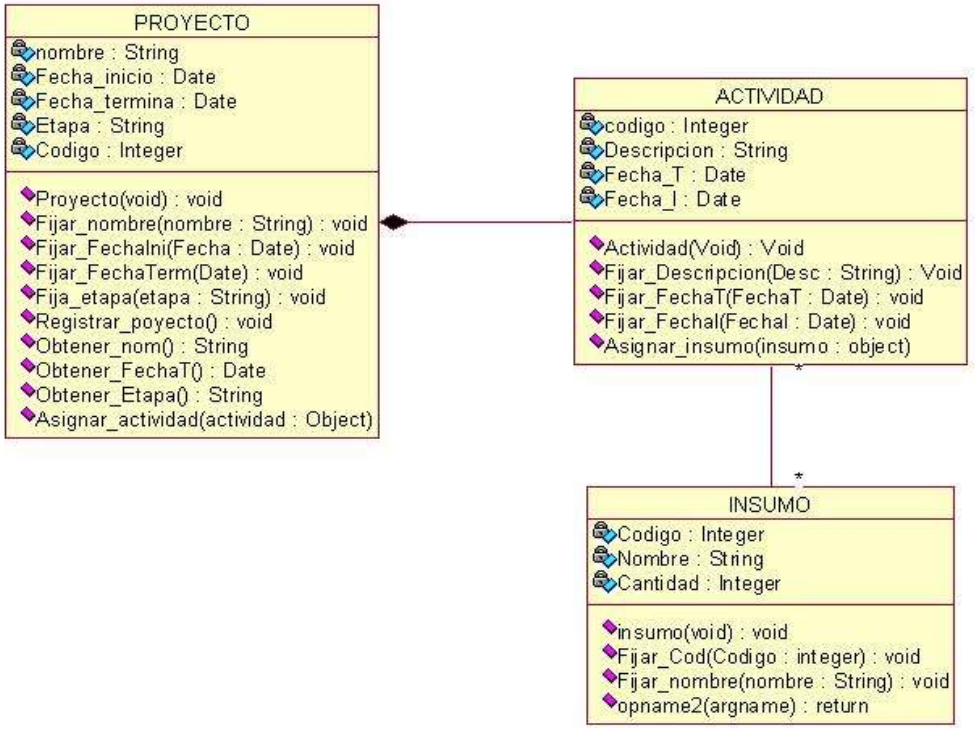
UML
DIAGRAMA DE CASOS DE USO

- Busca plasmar los requerimientos funcionales del nuevo sistema para cada uno de los usuarios.



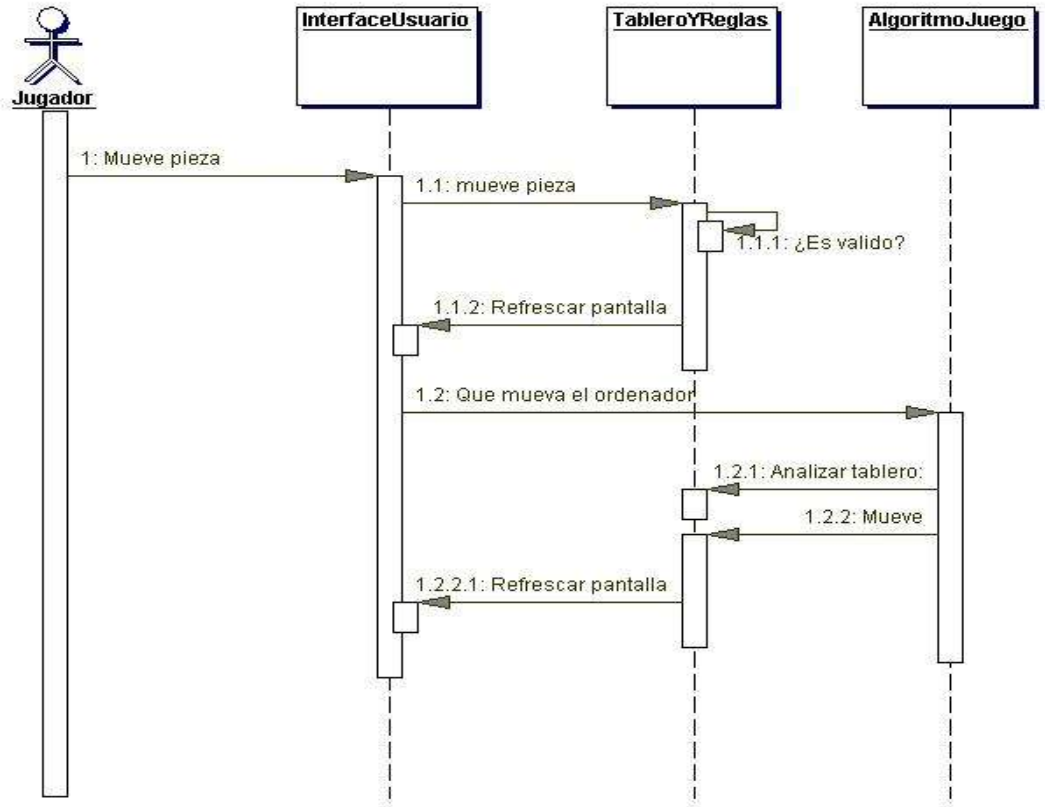
UML
DIAGRAMA DE CLASES

- Busca plasmar los elementos que intervienen en la aplicación ya sean los agentes que facilitan la comunicación con el usuario o los pertenecientes a la lógica del negocio. Describe los atributos y métodos de cada clase a implementar.



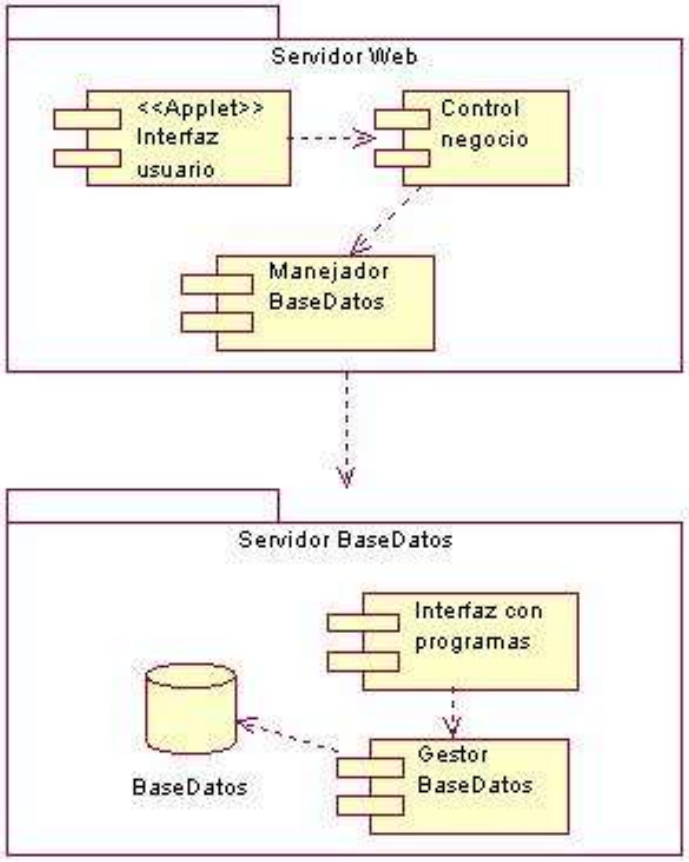
UML
DIAGRAMA DE SECUENCIA

- Permiten abstraer la secuencia de mensajes entre objetos a través del tiempo.



UML
DIAGRAMA DE DESPLIEGUE

- Integra los componentes Hardware y Software del sistema.



CLASES
VISUAL BASIC .NET

- Para crear una clase en visual Basic .NET debemos utilizar la palabra reservada **Class** seguida den nombre de dicha clase a construir:

```
Class Cliente
        instrucciones
End Class
```

- Por lo tanto, todo lo que se conforme como *instrucciones* estará encapsulado en la **Clase Cliente**.

CLASES
VISUAL BASIC .NET

- Creación de una clase Cliente con algunos elementos encapsulados:

```
Class Cliente
    Public Nombre As String
    Public Sub MostrarNombre()
        MsgBox("El nombre del cliente: ", Nombre)
    End Sub
End Class
```

- Para poder utilizar valores a las propiedades o variables que define la clase y además, utilizar los procedimientos y funciones encapsuladas debemos crear un **Objeto** Cliente:

```
Dim Cli As Cliente
Cli = new Cliente ( )
```

```
Dim Cli As new Cliente( )
```

OBJETOS
VISUAL BASIC .NET

- La clase:

```
Class Cliente
    Public Nombre As String
    Public Sub MostrarNombre()
        MsgBox("El nombre del cliente: “ & Nombre)
    End Sub
End Class
```

- El objeto: `Dim Cli As new Cliente( )`

- Para acceder a las instrucciones mediante el objeto :

```
Cli.Nombre = “Cesar Fernandez”
Cli.MostrarNombre ( )
```

HERENCIA Y POLIMORFISMO VISUAL BASIC .NET

- Las clases:

```
Class Cliente
    Public Nombre As String
    Public Sub MostrarNombre()
       MsgBox(Nombre)
    End Sub
End Class
```

```
Class ClienteMoroso
      Inherits Cliente  '→ HERENCIA
      Public Deuda As Decimal
End Class
```

- La clase: **ClienteMoroso** hereda los atributos y métodos públicos de la clase **Cliente.**

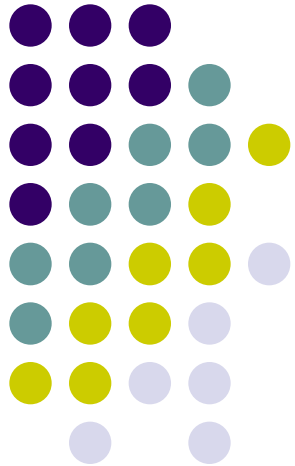
```
Dim Cli As new Cliente( )
Dim CliM As new ClienteMoroso( )
Cli.Nombre = "Cesar David"
CliM.Nombre = "Juan Jose"
Cli.MostrarNombre( )
CliM.MostrarNombre( )
```

→ **POLIMORFISMO**

MUCHAS GRACIAS

PROXIMAMENTE PROGRAMACION ORIENTADA A OBJETOS

INQUIETUDES O DUDAS ?



Ing. Cesar David Fernández Grueso.
CENTRO DE TELEINFORMATICA Y PRODUCCION INDUSTRIAL
SENA REGIONAL CAUCA