Universidad de Valparaíso
Facultad de Ingeniería

Escuela de
Ingeniería Informática

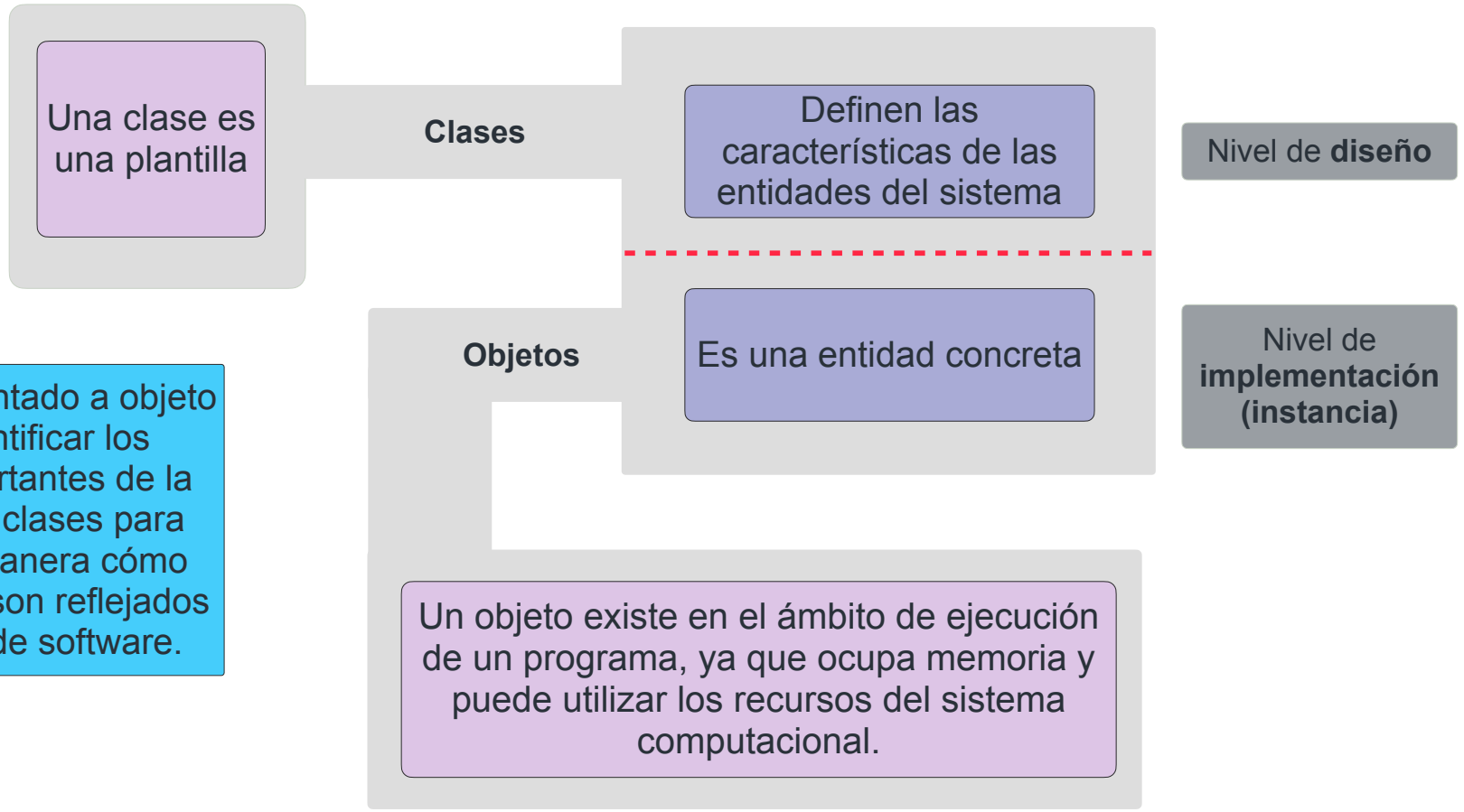
Programación orientada a objetos

Universidad de Valparaíso
CHILE

Programación orientada a objetos: una visión general

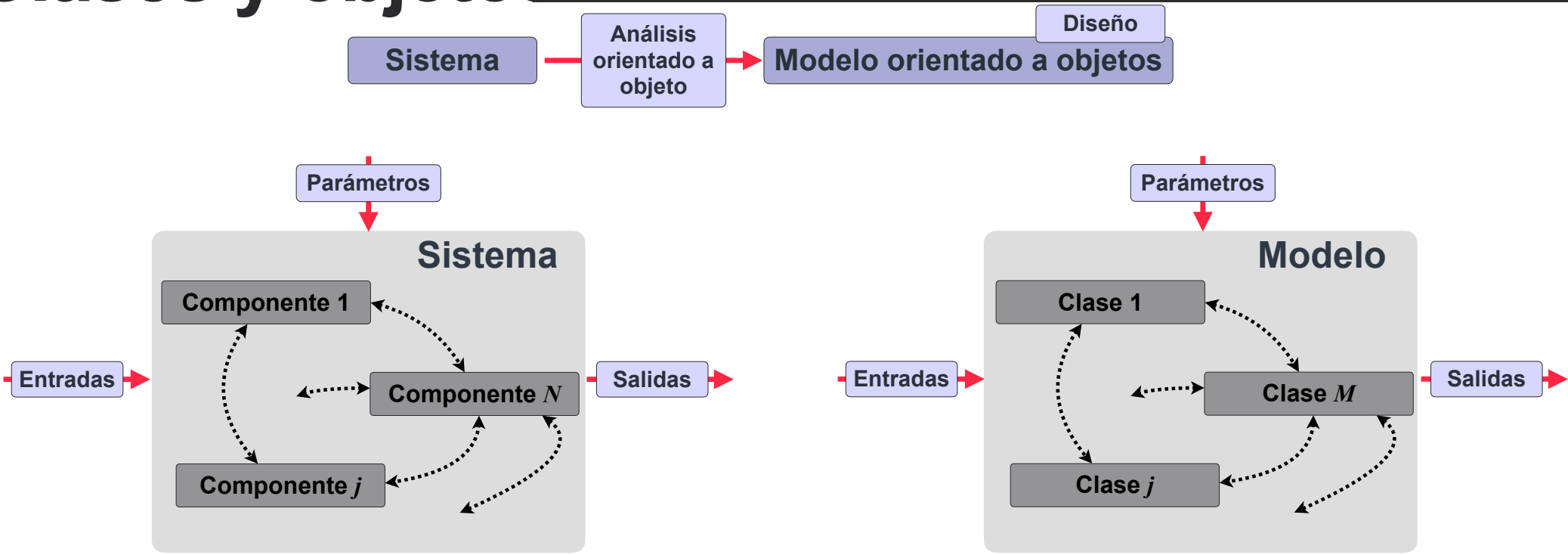
Clases y objetos

Universidad de Valparaíso CHILE



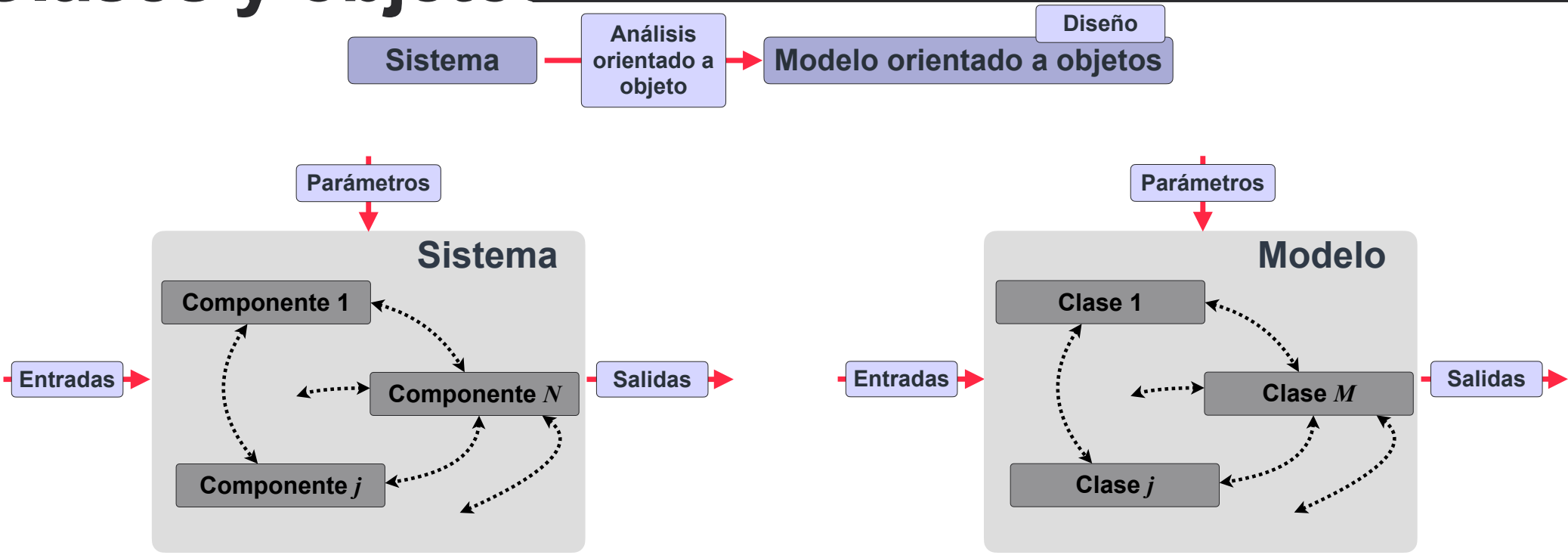
El modelado orientado a objeto involucra identificar los conceptos importantes de la solución y usar clases para estructurar la manera cómo esos conceptos son reflejados en un sistema de software.

Clases y objetos



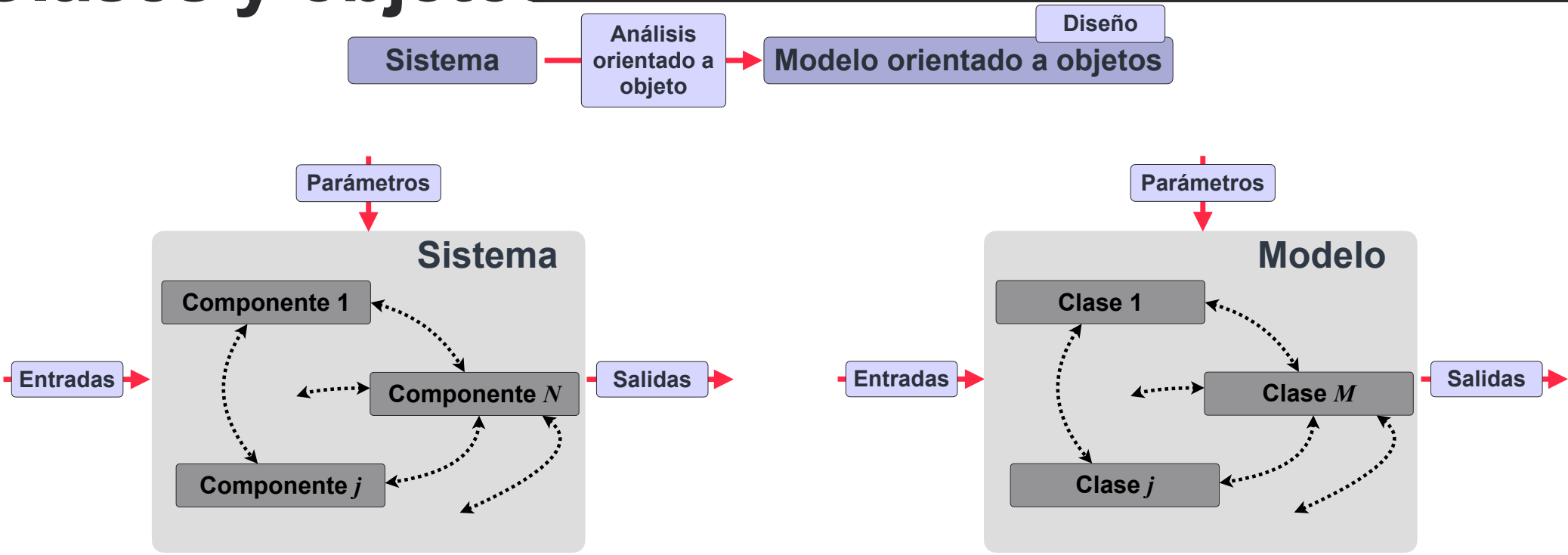
El modelado orientado a objeto involucra identificar los conceptos importantes de la solución y usar clases para estructurar la manera cómo esos conceptos son reflejados en un sistema de software.

Clases y objetos



Durante la fase de diseño se refinan las clases del modelo. Pueden descartar clases necesarias o agregar clases que no encontraron en el problema, pero que son útiles para e modelo. También pueden descartar, agregar o actualizar operaciones.

Clases y objetos



Los cambios realizados durante el análisis y el diseño son relativamente económicos (se cambian algunos diagramas y quizás algo de documentación) en comparación con los cambios realizados después de la programación o la implementación.

Universidad de Valparaíso
CHILE

Abstracción y encapsulación

Abstracción y encapsulación

Universidad de Valparaíso CHILE

Extracción de características esenciales de algo, ignorando los detalles que no aportan.

Oculta los detalles de implementación de los datos y proporciona una interfaz clara y definida para interactuar con ellos.

Abstracción

Objetos

Qué es lo que hace
(Cómo se utilizan)

Modelamiento

Encapsulación

Cómo se almacenan y manipulan

Nivel de **implementación**

los detalles de implementación pueden cambiar sin afectar a las partes del programa que interactúan con los datos a través de la interfaz definida.

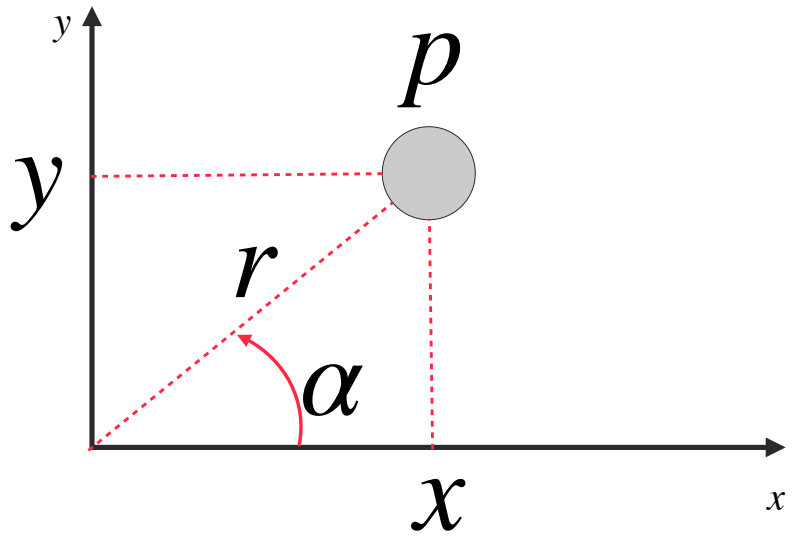
modularidad

reutilización del código

Ejemplo 1

Universidad de Valparaíso CHILE

Abstracción de un punto en $\mathbf{R}^2$

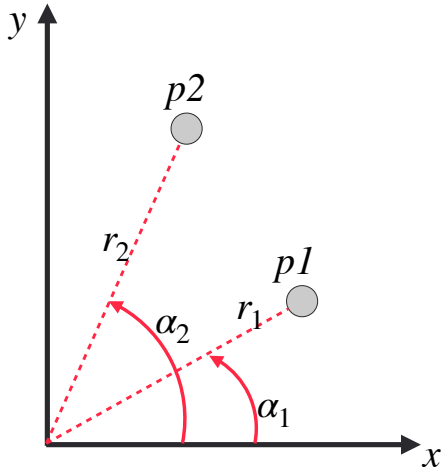


Ejemplo 1

```
CLASS Point:
    DOUBLE x,y
    DOUBLE r,alpha

    INIT_CLASS(x0,y0)
    DESTROY_CLASS()

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```

Una clase tiene variables internas que permiten mantener el estado cuando esté en "modo ejecución"

Atributos



Desde el punto de vista del modelo del sistema, los atributos son las variables de estado de un componente.

Ejemplo 1

```
CLASS Point:
    DOUBLE x,y
    DOUBLE r,alpha

    INIT_CLASS(x0,y0)
    DESTROY_CLASS()

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```

```
CLASS Point:
    STRUCT {
        DOUBLE x,
        DOUBLE y
    } rectangular

    STRUCT {
        DOUBLE r,
        DOUBLE alpha
    } polar

    INIT_CLASS(x0,y0)

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```

Distintas implementaciones que guardan las características de la clase.

Se deben ocultar a la vista del sistema que usa la clase **(encapsulación)**

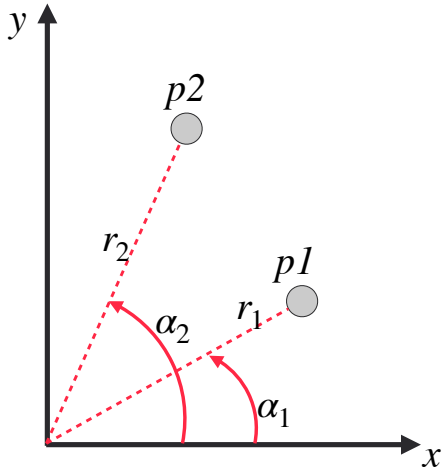
Ejemplo 1

```
CLASS Point:
    DOUBLE x,y
    DOUBLE r,alpha

    INIT_CLASS(x0,y0)
    DESTROY_CLASS()

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```

Una clase debe tener funciones que implementan la capa de **abstracción y encapsulación.**

Métodos

Los métodos actualizan los atributos según la dinámica de la solución modelada.

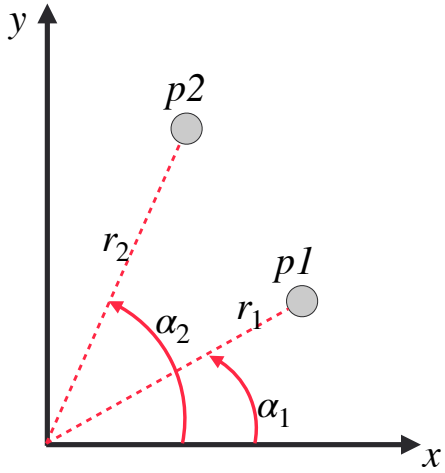


Ejemplo 1

```
CLASS Point:
    DOUBLE x,y
    DOUBLE r,alpha

    INIT_CLASS(x0,y0)
    DESTROY_CLASS()

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```

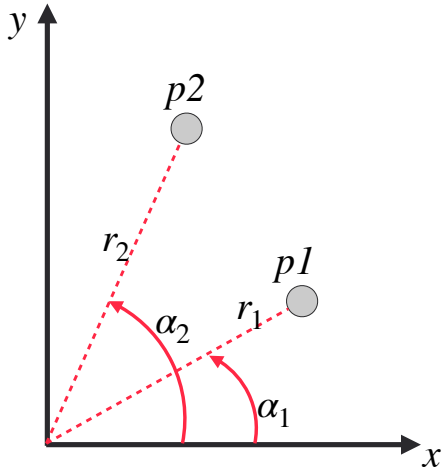
Constructor

Cuando se instancia una clase, SIEMPRE se ejecuta un método que permite **configurar** el objeto



Ejemplo 1

```
CLASS Point:
    DOUBLE x,y
    DOUBLE r,alpha

    INIT_CLASS(x0,y0)
    DESTROY_CLASS()

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```



Destructor

Cuando un objeto destruye, se llama a un método cuyo objetivo es liberar los recursos que el objeto pudo haber adquirido durante su vida útil.

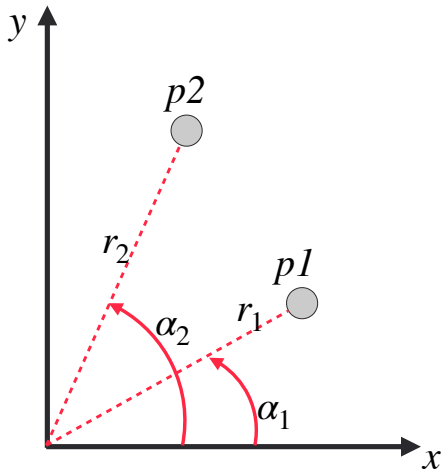
memoria dinámica
archivos abiertos
conexiones de red
...

El objeto deja su ámbito

Ejemplo 1

```
CLASS Point:
    DOUBLE x,y
    DOUBLE r,alpha

    INIT_CLASS(x0,y0)
    DESTROY_CLASS()

    DOUBLE angle()
    DOUBLE distanceToOrigin()
    DOUBLE distanceTo(Point p)
```

Creación de los objetos p1 y p2 a partir de la clase Point




```
BEGIN:

    Point p1 = MAKE_OBJECT Point(10.2,30.98)
    Point p2 = MAKE_OBJECT Point(-35.45, 60.66)

    DOUBLE r1 = p1.distanceToOrigin()
    DOUBLE d  = p2.distanceTo(p1)

END
```

Al terminar el ámbito, se llaman automáticamente a los destructures de los objetos creados en el ámbito

Invocar los métodos de la clase pero con las variables de estado del objeto

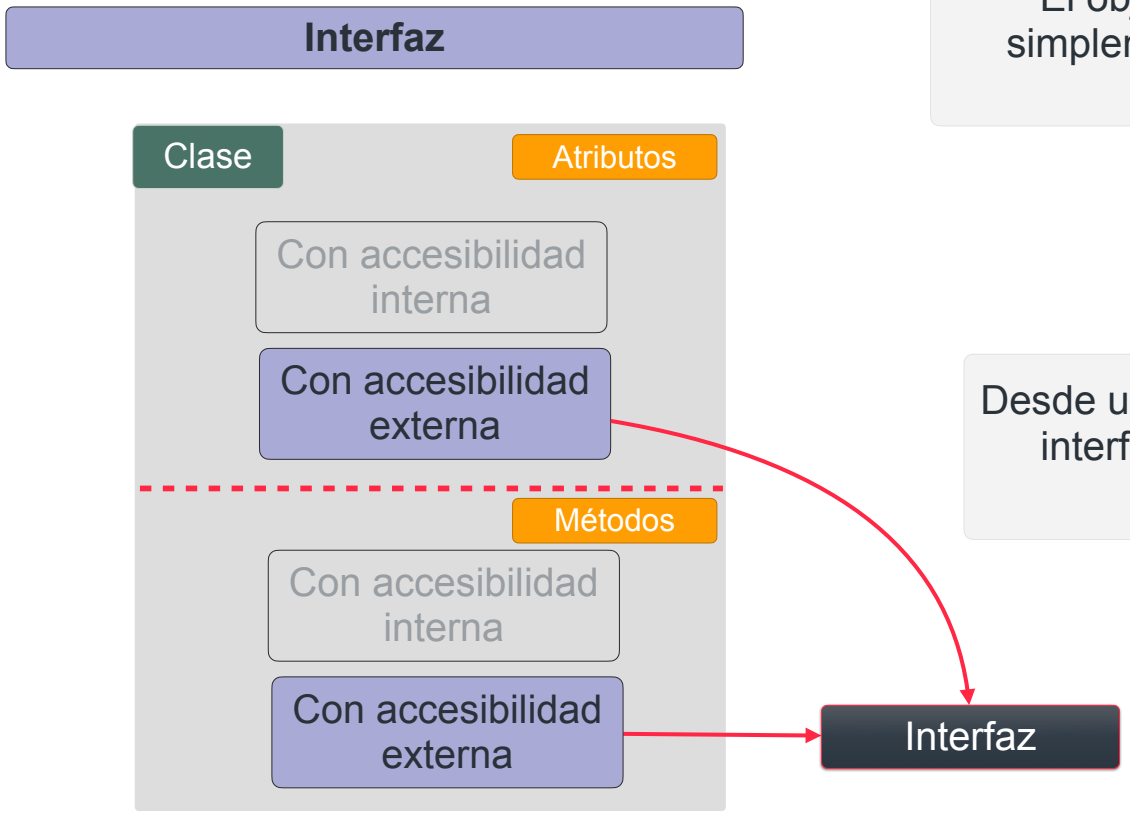
Ejemplo 2

Realizar la abstracción de un microbus

Universidad de Valparaíso
CHILE

Interfaz de una clase

Interfaz de una clase



El objeto se puede utilizar simplemente "conociendo" su interfaz.

No necesita conocer las características privadas o cómo las funciones realizan sus tareas.

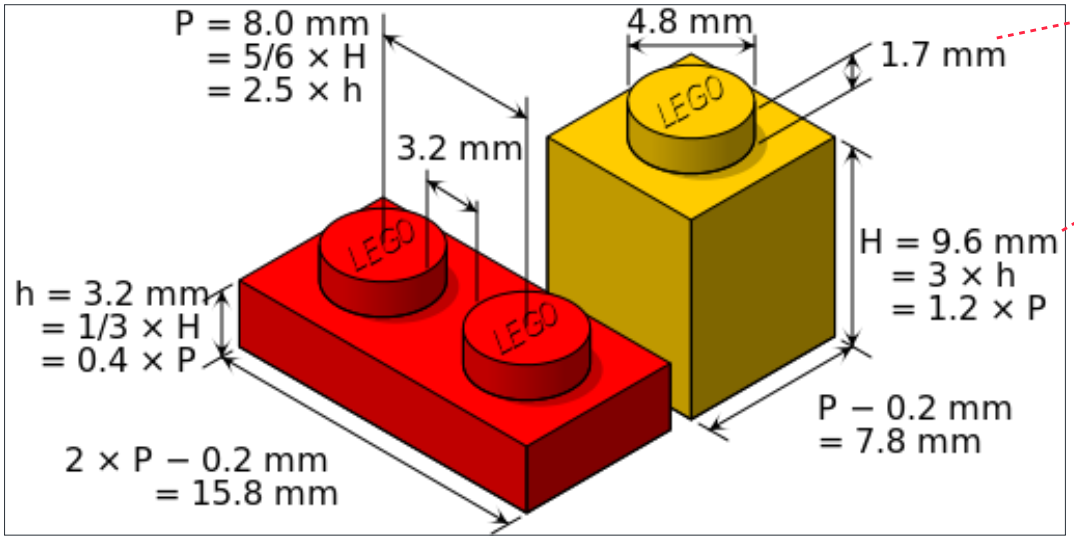
Desde un punto de vista de interconexión de sistemas, la interfaz es la forma en que un sistema se conecta, interactúa o utiliza otro sistema.

La Interfaz se implementa a través de atributos y métodos **públicos**

Interfaz de una clase

Universidad de Valparaíso CHILE

Ejemplo atributos privados



Los tamaños y el espaciado de los pernos y conectores están especificados con precisión

Ejemplo Interfaz

Haga cualquier cosa