

Programación Avanzada ***POO***

Clase 2 29/08/2108

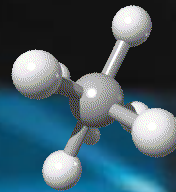
Lic. Víctor R. Pezantes

OBJETIVO:

Analizar y comprender el concepto de “POO”, y sus principales características

Objetivos Específicos:

- 1.- Entender el concepto de POO*
- 2.- Identificar las principales características de la POO*
- 3.- Entender el concepto de Herencia.*
- 4.- Entender el concepto de Polimorfismo.*
- 5.- entender el concepto de Encapsulación.*



TEMA: POO

Lic. Víctor R. Pezantes

Concepto de POO:

Forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos.

Concepto de POO:

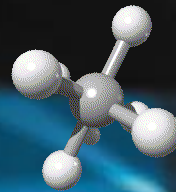
¿QUÉ ES UNA CLASE ?

Las clases son plantillas que agrupan comportamiento (métodos) y estados (atributos) de los futuros objetos.

¿QUÉ ES UN OBJETO?

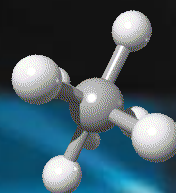
Los objetos son instancias de una clase

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real.



CARACTERÍSTICAS DE POO

Lic. Víctor R. Pezantes



HERENCIA

Lic. Víctor R. Pezantes

Concepto Herencia:



Analizar caso propuesto de Herencia:

Persona
(+)nombre:String (+)apellido:String (+)rut:String (+)edad:int
(+)caminar():void (+)hablar():void

Estudiante
(+)nombre:String (+)apellido:String (+)rut:String (+)edad: int
(+)caminar():void (+)hablar():void (+)aprender():boolean

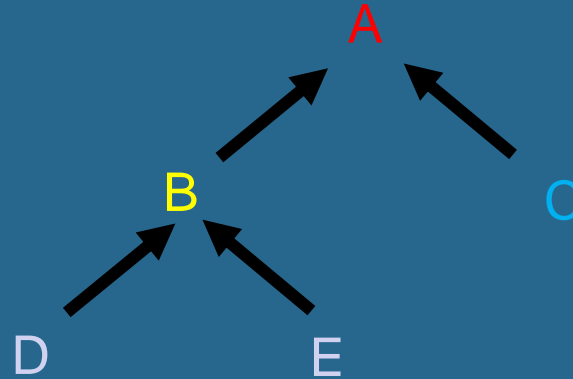
Docente
(+)nombre:String (+)apellido:String (+)rut:String (+)edad:int
(+)caminar():void (+)hablar():void (+)dictarClases():String

Identificar características principales de Herencia:

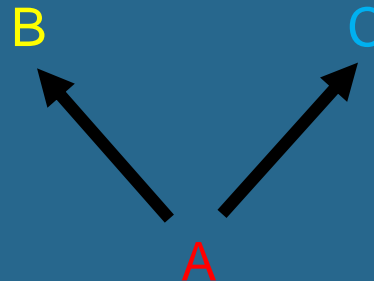
- 1.- Representar comportamiento en común.*
- 2.- Evitar duplicación de código.*
- 3.- Diseñar clases mas parecidas al mundo real.*

Identificar tipos de Herencia :

1.- Herencia simple.

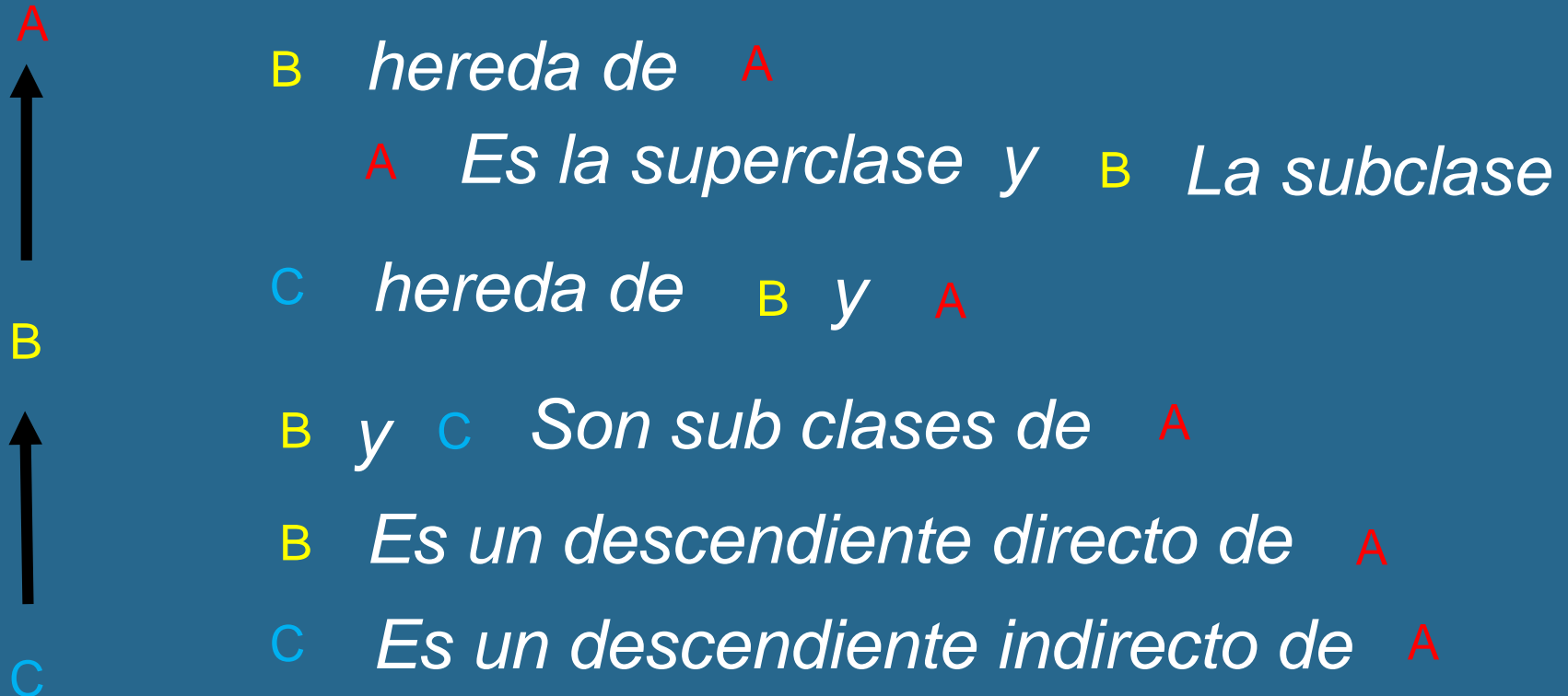


2.- Herencia múltiple.



Conocer el proceso transitivo de Herencia :

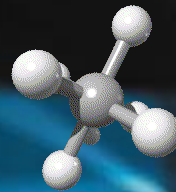
El proceso de herencia es transitivo



Identificar cuando no se puede heredar :

*En java se puede aplicar el modificador **final** a un método para indicar que no puede ser definido.*

*Asimismo, el modificador **final** es aplicable a una clase indicando que no se puede heredar de ella.*



POLIMORFISMO

Lic. Víctor R. Pezantes

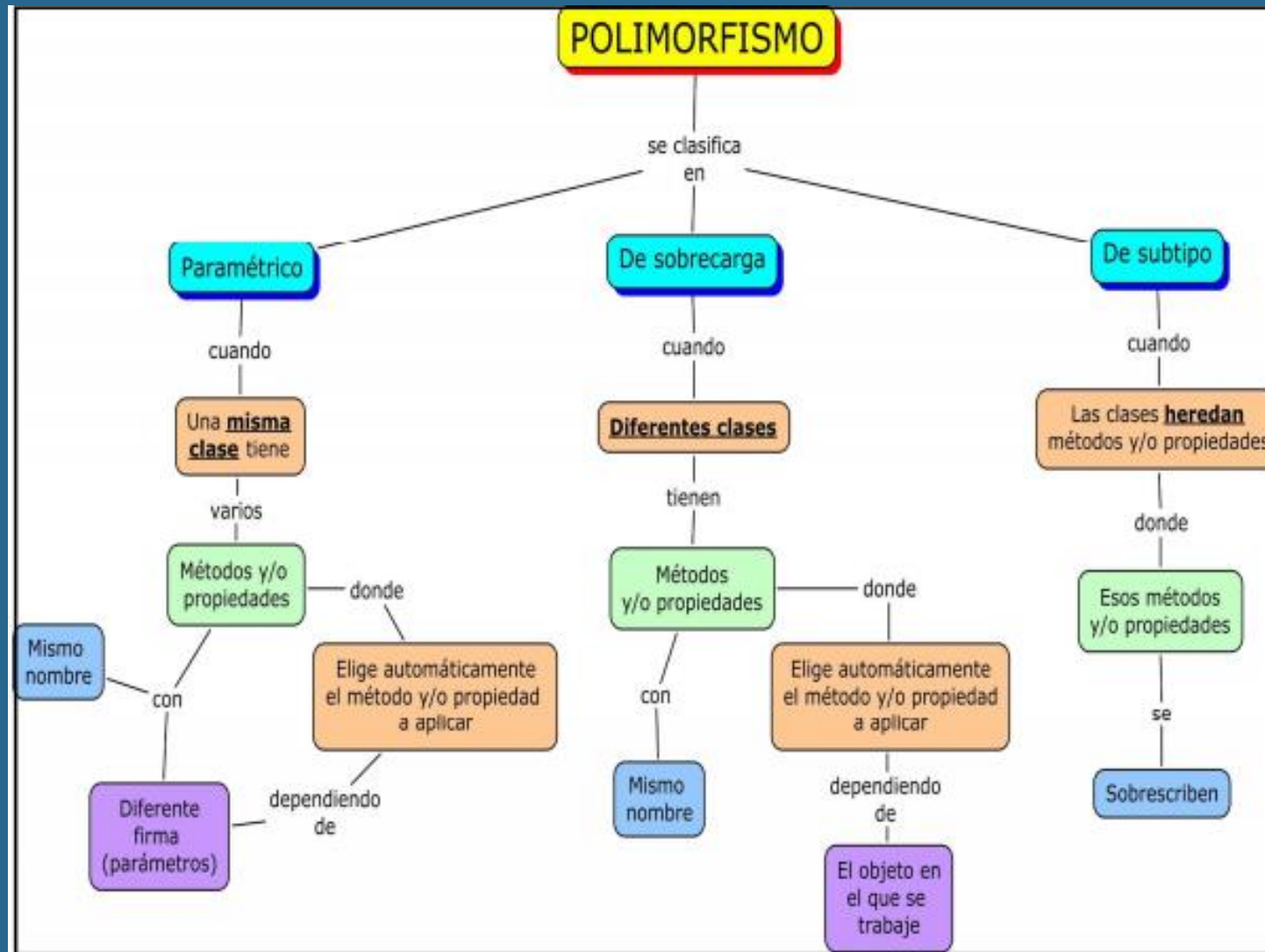
Concepto Polimorfismo:



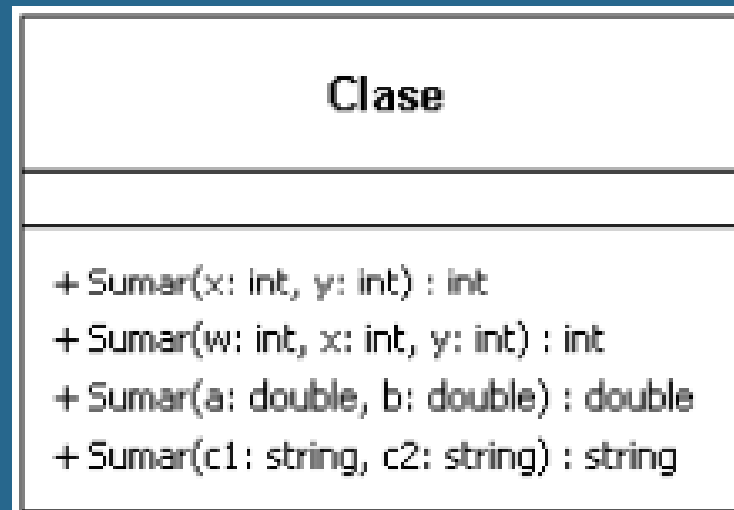
POLI = Múltiples **MORFISMO** = Formas

El polimorfismo se refiere a la posibilidad de definir múltiples clases con funcionalidad diferente, pero con métodos o propiedades denominados de forma idéntica, que pueden utilizarse de manera intercambiable mediante código cliente en tiempo de ejecución

Tipos de Polimorfismo:



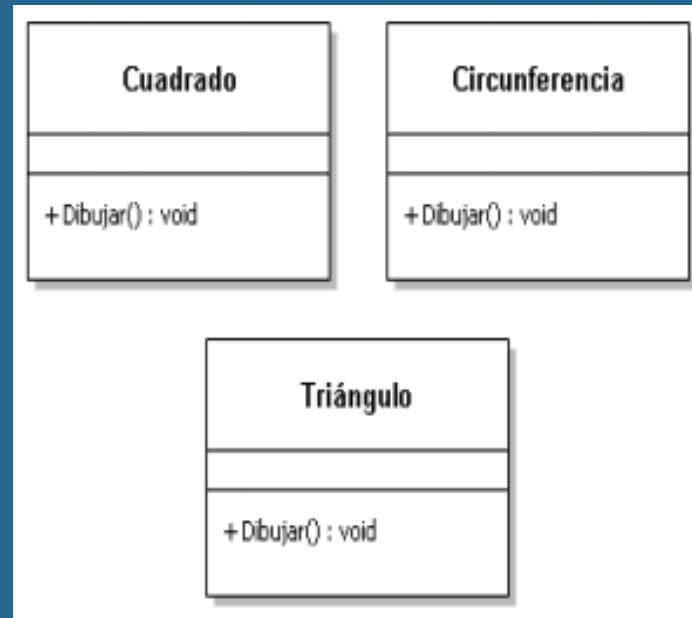
Polimorfismo paramétrico:



La misma clase tiene varios métodos con el mismo nombre pero diferentes firmas con diferentes tipos de datos

*La sobrecarga de métodos no provoca polimorfismo de **sobrecarga**, sino polimorfismo **paramétrico***

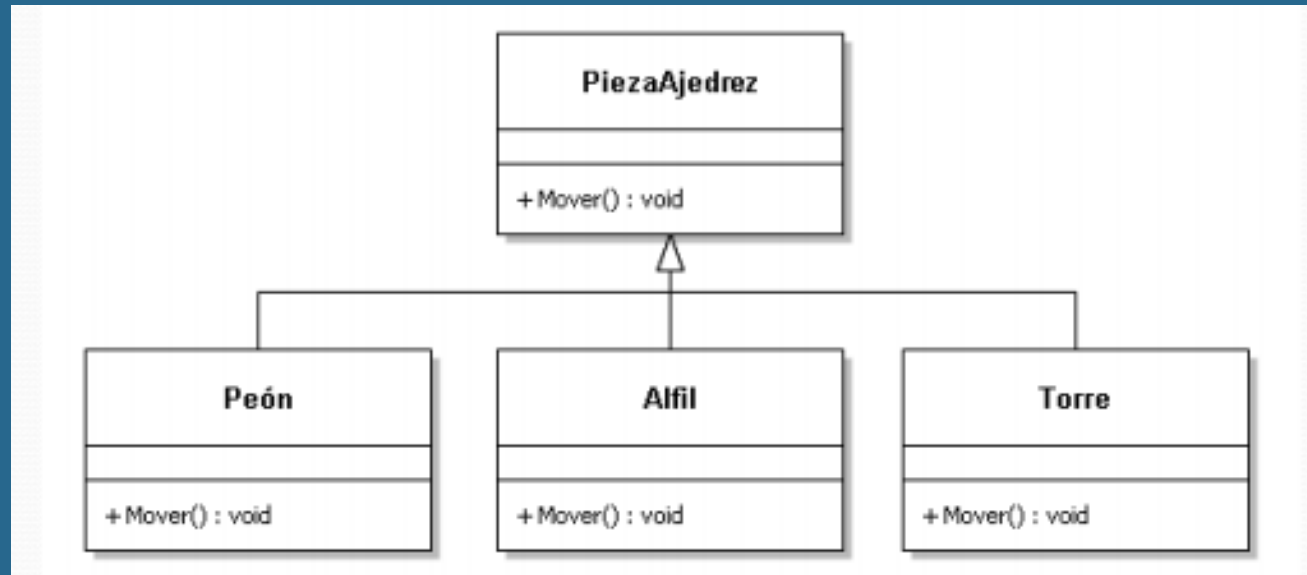
Polimorfismo Sobrecarga(overload):



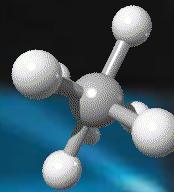
Diferentes clases tienen un método con el mismo nombre, pero comportamiento diferente

Se aplica el método de acuerdo al objeto en que se trabaje

Polimorfismo Subtipo(override):



Se sobrescribe el método heredado Mover() según lo requiera la pieza del ajedrez



ENCAPSULAMIENTO













Lic. Víctor R. Pezantes

Concepto encapsulación:



La encapsulación es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados. Por lo tanto, la encapsulación garantiza la integridad de los datos que contiene un objeto.

Niveles de acceso para el encapsulamiento:

Visibilidad	Public	Private	Protected	Default*
Desde la misma clase				
Desde una subclase				
Desde otra clase (no subclase)				

**Con default, me refiero a omitir el modificador de acceso.*

Métodos de acceso setters y getters:



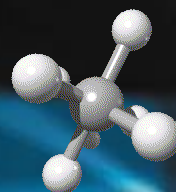
Los setters & getters son métodos de acceso lo que indica que son siempre declarados públicos, y nos sirven para dos cosas:

Setters:

Del Inglés Set, que significa establecer, pues nos sirve para asignar un valor inicial a un atributo, pero de forma explícita, además el Setter nunca retorna nada (Siempre es void), y solo nos permite dar acceso público a ciertos atributos que deseemos el usuario pueda modificar.

getters:

Del Inglés Get, que significa obtener, pues nos sirve para obtener (recuperar o acceder) el valor ya asignado a un atributo y utilizarlo para cierto método.



PREGUNTAS

????

Lic. Víctor R. Pezantes



Lic. Víctor R. Pezantes