

Metodologías de Programación I

Introducción a Objetos



Docentes Teoría


- Lic Balbi M Luciana ibalbi@unnoba.edu.ar
- Lic Di Cicco Carlos carlosd Cicco@unnoba.edu.ar

Docentes Prácticas

- Lic Naso Federico federico.naso@gmail.com
- Mg Di Grazia Nelson nelson.digrazia@telefonica.com

Docente Consulta

Sebastián Sottile sebastiansottile@unnoba.edu.ar

- 
- Motivación
 - TADs
 - Intro OO
 - Relacion TADs -POO
 - Definiciones básicas
 - Objetos
 - Mensajes y métodos
 - Clases, subclases y objetos
 - Herencia
 - Conceptos claves
 - Encapsulamiento
 - Abstracción
 - Polimorfismo

Tipo Abstracto de Datos

- Es la noción matemática que define un tipo de datos.
- Esta descripción abstracta y formal se especifica mediante la especificación sintáctica, donde defino nombre, dominios, y rango de operaciones sobre el tipo y la especificación semántica donde por un conjunto de axiomas establezco como opera cada operación
- Por lo tanto un TAD se define por un numero de operaciones aplicables, el modo que puede invocarse cada operación (sintaxis) y sus efectos (semántica).

Ejemplo TAD Natural

Nombre: Natural

Dato: secuencia de dígitos precedida por un signo positivo.

Operaciones:

Crear \rightarrow Natural

$=: (\text{Natural}, \text{Natural}) \rightarrow \text{Booleano}(\text{verdadero}, \text{falso})$

$>: (\text{Natural}, \text{Natural}) \rightarrow \text{Booleano}(\text{verdadero}, \text{falso})$

$+: (\text{Natural}, \text{Natural}) \rightarrow \text{Natural}$

....

Axiomas y Operaciones

0(cero) es un numero natural.

Si n es un numero natural, también lo es $S(n)$, su siguiente.

$(S(n) = 0) = \text{falso}$

$(n + 0) = n$

si $S(n) = S(m) \Rightarrow n = m$

....

Sintaxis

Semántica

Tipo Abstracto de Datos

Todo TAD debe cumplir con 2 ppios:

- Ocultación de la info (encapsulación) : solo puede accederse a los valores del tipo que define o modificarlos mediante las operaciones abstractas definidas sobre ellos. La encapsulación hace referencia al ocultamiento de la info y a la capacidad para expresar al unidad formada por los valores y las operaciones
- Abstracción de Datos: se separan las propiedades lógicas de los datos de su representación o implementación

Tipo Abstracto de Datos

- Cualquier implementación software de un TAD debe seguir cumpliendo los ppios de ocultación de la información y abstracción de datos.
- Así, la implementación deberá ocultar la representación del tipo (su estructura de datos) y solo podrá accederse a sus valores (datos) mediante las funciones (interfaz)
- Un TAD especifica el qué, no el cómo, un tipo predefinido especifica el cómo basándose en el qué, y lo oculta al programador.

Tipo Abstracto de Datos

- TADs permiten la creación de instancias con propiedades y comportamiento bien definidos. En orientación a objetos TADs son referidos como *clases*. Por lo tanto una clase define las propiedades de objetos, los que son instancias en un entorno orientado a objetos.



Qué vimos hasta ahora?

Programación Estructurada

La idea principal de esta forma de programación es separar las partes complejas del programa en módulos, que se ejecuten a medida que sea necesarios.

Estos módulos son independientes entre sí, y además deben poder comunicarse.

Introducción

Problemas de la Programación Estructurada


Varios programadores trabajan en equipo desarrollando una aplicación grande.

Más de un programador manipula funciones separadas que pueden referirse a tipos de datos mutuamente compartidos, y los cambios de un programador se deben reflejar en el trabajo del resto del equipo.

Qué pasa si uno de los programadores decide que una estructura existente en el sistema en vez de representarse con una lista, ahora se representa con un arreglo?????

Este fue uno de los problemas de la programación estructurada, por lo cual se siguió investigando sobre diferentes metodologías de programación.


Programación Orientada a Objetos



Según Booch:

La programación Orientada a Objetos es un método de implementación en el cual los programas están organizados como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son todas miembros de una jerarquía de clases unidas vía relaciones de herencia.

Diseño Orientado a Objetos



Diseño orientado a objetos es un método de diseño que guía el proceso de descomposición orientado a objetos y define una notación para expresar tanto los modelos lógico (estructura de clase y objeto) y físico (arquitectura de módulo y proceso) (tanto estáticos como dinámicos).

Paradigmas de programación.

- Def 1:

Un paradigma es un marco de referencia que impone reglas sobre cómo se deben hacer las cosas, indica qué es válido dentro del paradigma y qué está fuera de sus límites. Un paradigma distinto implica nuevas reglas, nuevos elementos, límites y maneras de pensar, implica un cambio, y todo cambio es difícil.

- Def 2:

Un estilo o paradigma de programación es una manera de organizar programas sobre la base de algún modelo conceptual de programación y de un lenguaje apropiado para hacer que los programas sean escritos de una manera clara.

Paradigmas de programación.

Definamos entonces como primer axioma que trabajar con objetos es trabajar con un paradigma nuevo. Esto implica que debemos definir cuáles son las reglas de este nuevo paradigma.

- La primera regla del paradigma trata sobre qué se intenta obtener como resultado del desarrollo de sistemas utilizando objetos. Esta es: “Un sistema hecho con objetos es un modelo computacional de una porción de la realidad (la porción que queremos sistematizar, denominada dominio)”.

- Qué es lo que ves?



- Qué es lo que ves?



- Qué es lo que ves?



Qué es lo que tienen en común?

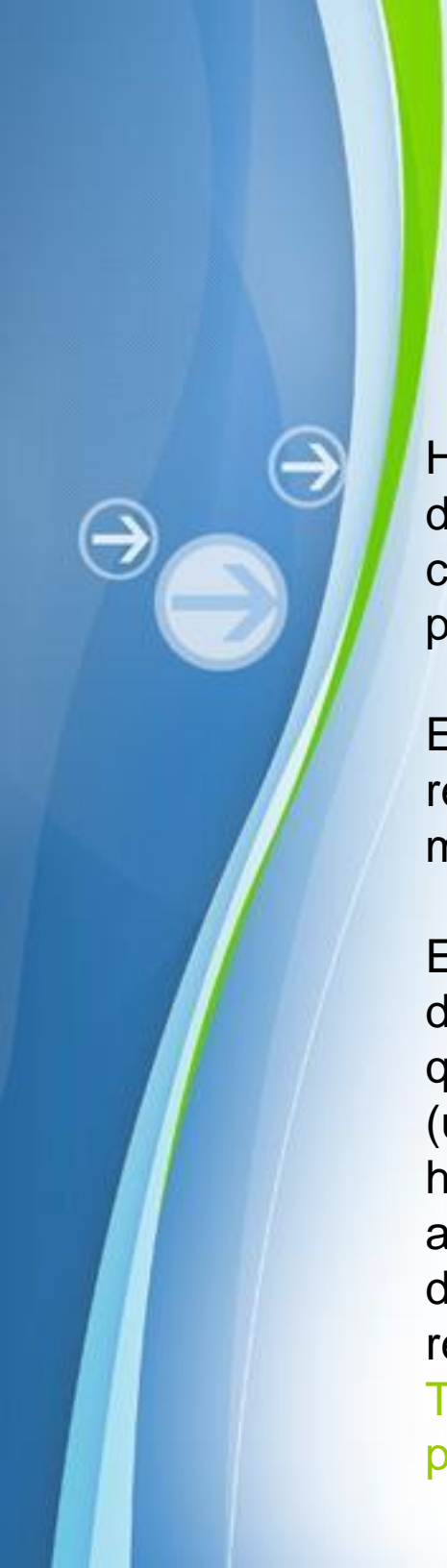


Modelo
MArca
Velocidad
Color

Acelerar
Desacelerar
Apagar
Arrancar

Se podría encontrar una forma de definir “algo” que encapsule las características y comportamiento comunes

Ejemplo




Hay una situación de la realidad ,por ejemplo facturar una venta, que debe ser modelada de tal manera que pueda ser ejecutada por una computadora, obteniendo como mínimo el mismo resultado que el proceso reemplazado por este nuevo sistema producía.

En nuestro ejemplo del proceso de facturación, un elemento de la realidad es la **factura**, otro es la **persona** que realiza la factura y no menos importante el motivo por el cual se está facturando: la **venta**.

En nuestro modelo por lo tanto, debemos representar estos elementos de la realidad con objetos. Hay elementos fáciles de reconocer puesto que son elementos físicos como “la factura” o la persona que factura (una persona que durante este proceso asume el rol de “facturador”) y hay elementos más difíciles de ver puesto que son elementos abstractos, ideas o acciones, como la venta en sí misma, que también debe ser modelada puesto que una factura es el resultado de haber realizado “una venta”.

Tenemos que ocuparnos de “modelar bien” la realidad de nuestro problema.

Ejemplo

- 
- Debemos ocuparnos de las “**responsabilidades**” de estos objetos, de **qué** hacen.
 - Pensemos en la responsabilidad de “un facturador”. ¿Qué hace un facturador? Su responsabilidad es “facturar”. La responsabilidad de una factura no será más que permitir saber cuál es su número, qué dicen las líneas que la componen o el total de la misma (recuerden que una factura es simplemente un papel con datos impresos). La responsabilidad de “la venta” será indicar qué se vendió, a qué precio y a quién.
 - Cada vez que se agrega una responsabilidad a un objeto debemos estar seguros que realmente corresponda a ese objeto. Por ejemplo, ¿debe “la factura” tener cómo responsabilidad “imprimirse”? Pensemos en la realidad. Una factura no se imprime “a sí misma”, alguien se encarga de imprimirla, por lo tanto “la factura” no tiene la responsabilidad de “imprimirse”. Menos aún de “guardarse” o de “leerse” de un archivo. Están son responsabilidades que le corresponden a otros objetos que también deben ser modelados

Programación Orientada a Objetos

- Podríamos decir por ahora que desarrollar con objetos es modelar el problema que tenemos que resolver utilizando objetos. La realidad tiene ciertas características interesantes que debemos poder simular si queremos tener un buen modelo. Una de esas características es la interacción entre los elementos de la realidad. Por ejemplo, el gerente de contabilidad le pide al facturador que facture y el facturador le indica a la factura a quién está dirigida. Estos elementos “colaboran” entre sí para llevar adelante el proceso de facturación.
- Esta interacción se modela en el paradigma de objetos por medio de “mensajes”. Los objetos se comunican entre sí utilizando “mensajes” (que nada tienen que ver con mails, mensajes de red, etc.) que representan la comunicación o interacción de elementos de la realidad.
- Como todo proceso de comunicación hay un emisor y un receptor. Por lo tanto el objeto emisor le indica por medio de un mensaje al objeto receptor qué debe hacer.
Definimos entonces que un programa en el paradigma de objetos es “un conjunto de objetos que colaboran entre sí enviándose mensajes”

Programación Orientada a Objetos

- Otra característica de la realidad consta en definir las responsabilidades más allá de cómo se llevan a cabo. Al gerente de contabilidad no le interesa si Juan (el facturador) factura escribiendo primero a quién va dirigida la factura y luego el detalle de la misma. La manera en que Juan factura no le importa, siempre y cuando realice correctamente su trabajo. Esta característica permite al gerente de contabilidad despreocuparse sobre cómo se está realizando la facturación y ocuparse de sus propias tareas.
- Esta característica en el paradigma de objetos se llama “encapsulamiento” y es la que permite separar qué hacen los objetos (responsabilidad) de cómo lo hacen (implementación).
- Por último, otra característica que debemos poder simular en nuestro modelo es delegar las responsabilidades a alguien sin preocuparnos quién sea ese alguien. Al gerente de contabilidad no le importa si el facturador es Juan o Pedro, le importa poder pedirles que facturen y que la facturación se realice. Esa es la responsabilidad del facturador.

Programación Orientada a Objetos

- Esta característica es muy importante, permite despreocuparse sobre quién hace el trabajo y solamente asegurarse de tener buenos “colaboradores” en quienes delegar tareas. Debemos estar acompañados de colaboradores si queremos llevar adelante ciertas tareas, y lo que nos interesa de esos colaboradores es que cumplan con sus responsabilidades correctamente, más allá de cómo lo hagan.
- Esta característica de la realidad no puede escapar de un paradigma que se jacta de modelarla, y por supuesto no se escapó. Esta característica en objetos se denomina “polimorfismo” y justamente permite al objeto emisor del mensaje despreocuparse de quién es exactamente su colaborador, sólo le interesa que sea responsable de llevar adelante la tarea que le encomienda a través del “mensaje”.

Programación Orientada a Objetos

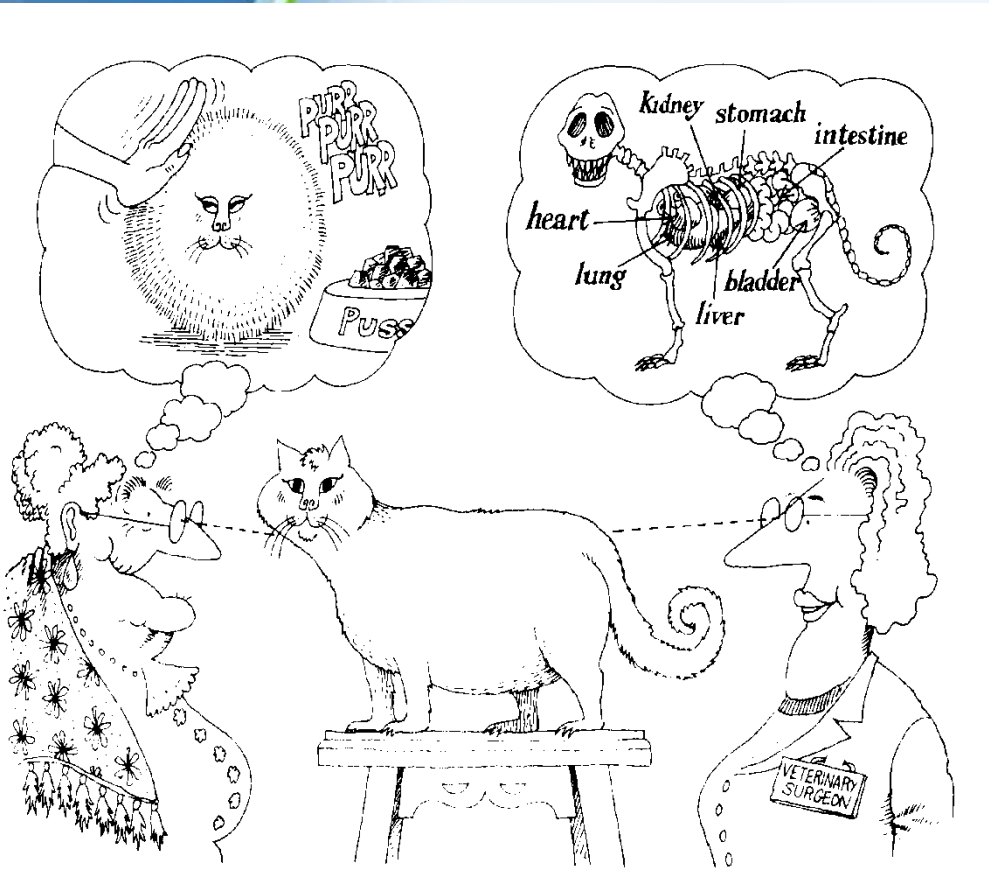
En el paradigma orientado a objetos, el marco conceptual es el *modelo de objetos*. Hay cuatro elementos principales en este modelo:

- [Abstracción](#)
- [Encapsulación](#)
- [Modularidad](#)
- [Jerarquía](#)

Veamos cada uno en detalle...

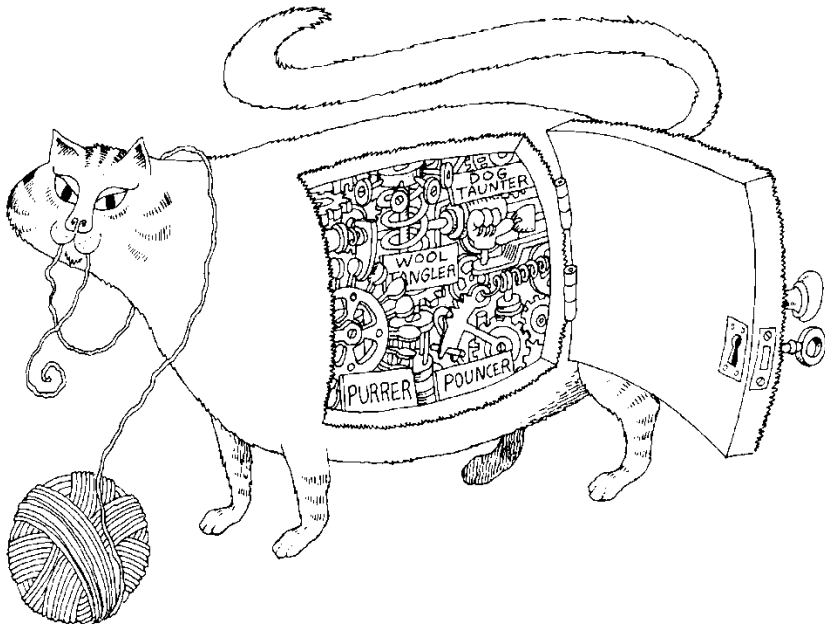
Abstracción

- Una abstracción es una descripción simplificada de un sistema que enfatiza algunos de sus detalles o propiedades mientras suprime otros. Una buena abstracción es la que enfatiza detalles que son significantes al lector, y suprime los que no lo son.
- **La abstracción enfoca las características esenciales de algún objeto, relativa a la perspectiva del observador**
- Una abstracción se enfoca sobre una vista externa del objeto, y sirve para separar su comportamiento esencial de su implementación.



Encapsulación

- La abstracción de un objeto debería preceder a su implementación. Una vez que la implementación es seleccionada, esta debería tratarse como un secreto de la abstracción y ocultársela a la mayoría de los clientes.
- Abstracción y encapsulamiento son conceptos complementarios: la abstracción enfoca la vista externa de un objeto, y el encapsulamiento (ocultamiento de la información) previene que los clientes vean la parte interna, donde el comportamiento de la abstracción es implementada.
- Según Booch: *"El encapsulamiento es el proceso de ocultar todos los detalles de un objeto que no contribuyen a sus características esenciales."*



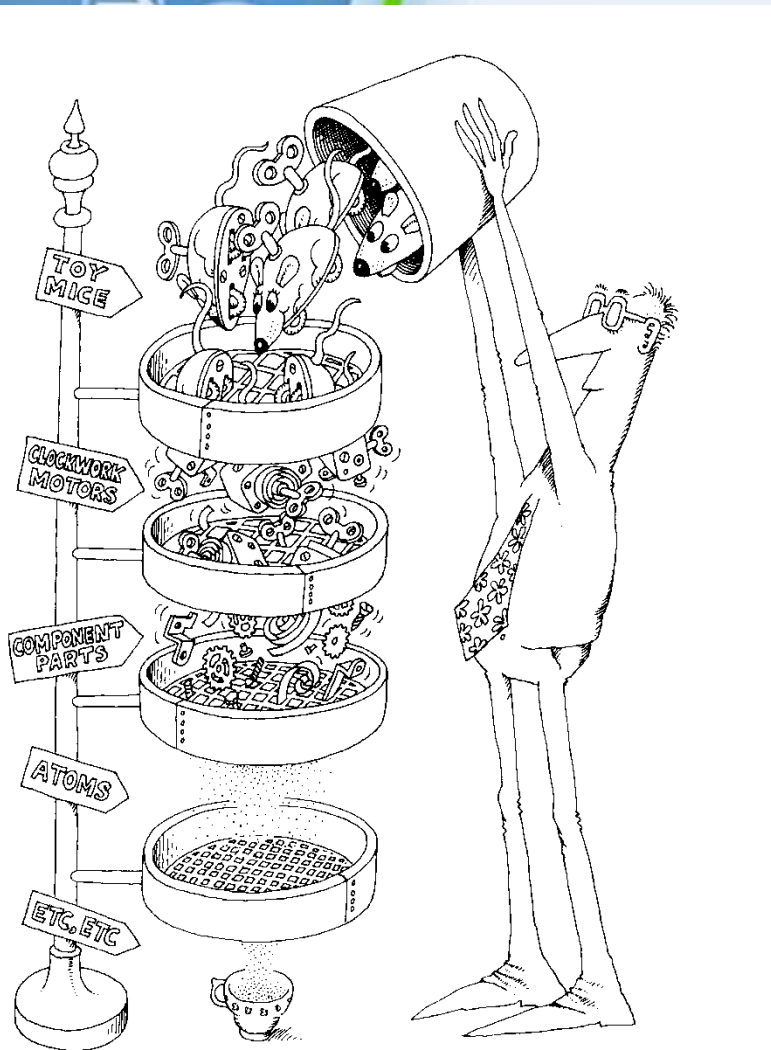
Modularidad

- Si bien el acto de particionar un programa en componentes individuales reduce en algún grado la complejidad, una razón más poderosa para realizarlo es que ésto crea un número de contornos bien documentados dentro del programa.



Jerarquía

- La abstracción es sin duda algo bueno, pero salvo en aplicaciones triviales, encontramos que hay mas abstracciones diferentes que las que podemos comprender a un dado momento. El encapsulamiento nos ayuda a manejar esta complejidad ocultando la parte interna de estas abstracciones. La modularidad también ayuda, dándonos una manera de agrupar lógicamente abstracciones relacionadas. Esto no es suficiente. Un conjunto de abstracciones frecuentemente forman una jerarquía, e individualizándolas en nuestro diseño, simplificaremos el entendimiento del problema.
- *La jerarquía es un ordenamiento de abstracciones.*
- Las dos jerarquías más importantes son: estructura de clases (jerarquía *tipo de*) y la estructura de objetos (jerarquía *parte de*).



Resumiendo

Desarrollar con objetos es modelar la realidad utilizando objetos que deben colaborar entre sí enviándose mensajes. Estos objetos sólo se preocupan de que sus colaboradores sepan qué hacer (responsabilidad), no cómo lo hacen, y no tienen problemas sobre quiénes son sus colaboradores (polimorfismo) siempre y cuando respondan sus mensajes.