Unidad 1
Introducción al Paradigma Orientado a Objetos

Prof: Lic. Rosales Pablo (prosales@unpata.edu.ar)
Materia: Programación Orientada a Objetos
Año: 2024

¿Qué es un paradigma?

¿Por qué este término es usado para describir el enfoque orientado a objetos para resolver un problema?

¿Cómo el lenguaje influencia el pensamiento?

Algunas características del pensamiento orientado a objetos.

¿Por qué POO?

Se mantiene hace años como el paradigma más popular, ¿por qué?

1. Su éxito está probado en el desarrollo de SW.
2. Permite escalar de problemas chicos a grandes.
3. Similar a técnicas de pensamiento para resolver problemas en otros dominios.

Paradigma

paradigma

Del lat. tardío *paradigma*, y este del gr. παράδειγμα *parádeigma*.

1. m. Ejemplo o ejemplar.

2. m. Teoría o conjunto de teorías cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento. *El paradigma newtoniano.*

3. m. *Ling.* Relación de elementos que comparten un mismo contexto fonológico, morfológico o sintáctico en función de sus propiedades lingüísticas.

4. m. *Ling.* Esquema formal en el que se organizan las palabras que admiten modificaciones flexivas o derivativas.

¿Qué tiene que ver esto con la programación?

Paradigma

Thomas Kuhn en “The Structure of Scientific Revolutions” **lo describe como un conjunto de teorías, estándares y métodos que juntos representan una forma de organizar el conocimiento, una forma de ver el mundo.**

Su tesis consistió en que las revoluciones en la ciencia ocurren cuando un paradigma es re examinado, rechazado y reemplazado por otro.

Robert Floyd en “The Paradigms of Programming” nos dice que un paradigma de programación es una **forma de conceptualizar que significa realizar una computación, de cómo estructurar y organizar las tareas que debe ejecutar una computadora**.

Hipótesis de Sapir-Whorf

En la lingüística, es una hipótesis que plantea que el lenguaje en el que se piensa una idea influye en la comprensión del mundo que nos rodea.

- Esquimales y la nieve.
- Árabes y los camellos.
- Palabras para colores y espacio.

¿Es posible algo así con lenguajes de programación?

Tesis de Church-Turing

“Todo algoritmo puede ser computado por una máquina de turing”.

Si lo damos por cierto, todo puede ser ejecutado por cualquier lenguaje, pero podria ser mas facil usar un lenguaje por sobre otro dependendiendo el problema.

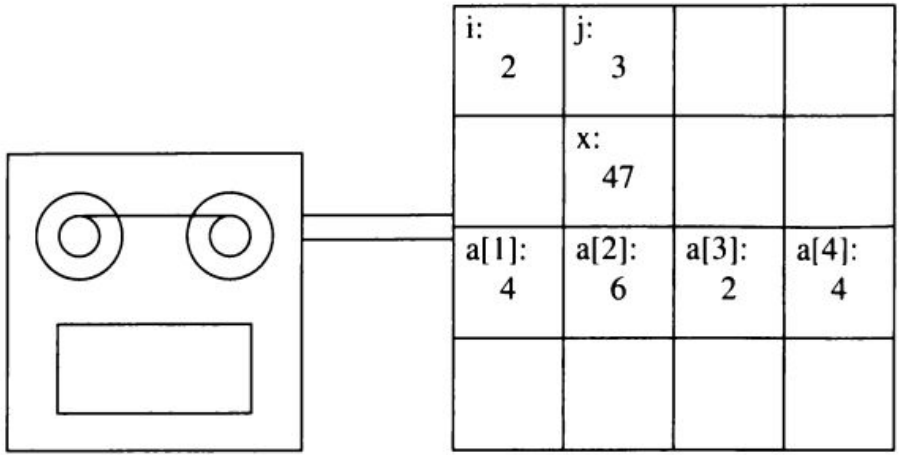
Los lenguajes de programación te orientan, pero no te impiden.

Modelo teórico que consiste en una cinta infinita y un cabezal lector-escritor de símbolos que sigue un conjunto de reglas predefinidas.
Fundamental para la teoría de computabilidad y demuestra la resolución de problemas mediante algoritmos.

Paradigma Imperativo

El “tradicional”, con:

- Estados
- Variables
- Asignaciones
- Bucles



Una unidad de procesamiento separado de la memoria, que procesa y modifica sobre la misma.

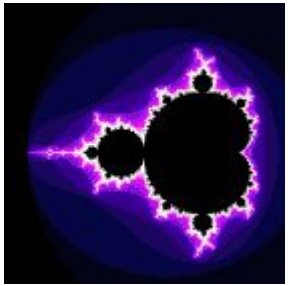
Paradigma Imperativo

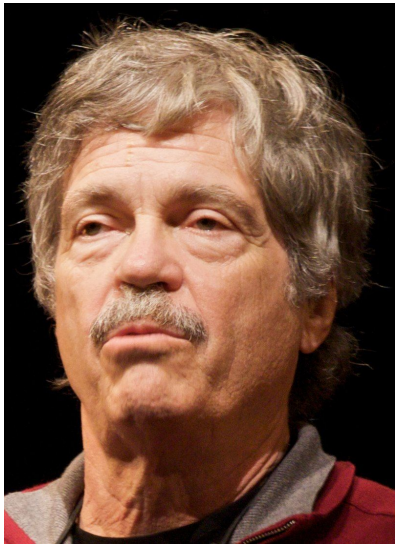
Es un paradigma extremadamente cercano a como el hardware computa en un nivel fundamental.

Lejano a cómo las personas resolvemos problemas. A mayor complejidad más difícil la solución.

- Complejidad en sistemas grandes.
- Dificultad de mantenimiento.
- Mala gestión de los estados.
- poca reusabilidad y abstracción.
- poca flexibilidad y adaptabilidad (ligados al HW subyacente).

Paradigma Orientado a Objetos





Alan Kay se preguntó: ¿Por qué construimos a partir de piezas que son inútiles por sí solas?

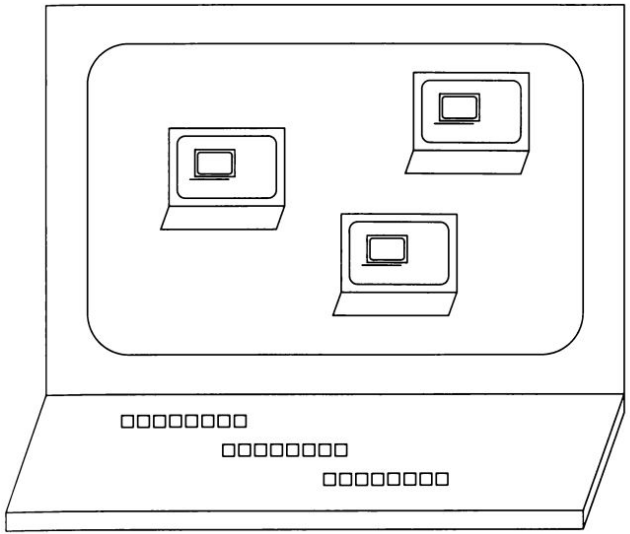
¿Por qué no construir con piezas similares a todo nivel de detalle? (fractales).

Otros aportes de Kay:

- Concepto de objeto.
- Smalltalk.
- metáfora del escritorio.
- Dynabook: concepto de notebook.

Diseño recursivo

En vez de ver al cómputo dividido en componentes primitivos (procedimientos y estructuras de datos) deberíamos verlo como un universo de piezas de computo conectadas interactuando entre sí. Cada una de estas llamadas “objetos”, como una computadora en sí mismo, compuesta de una unidad de procesamiento y memoria. Pero la memoria en este caso es otra vez constituida por otros objetos y así “hasta el infinito”.



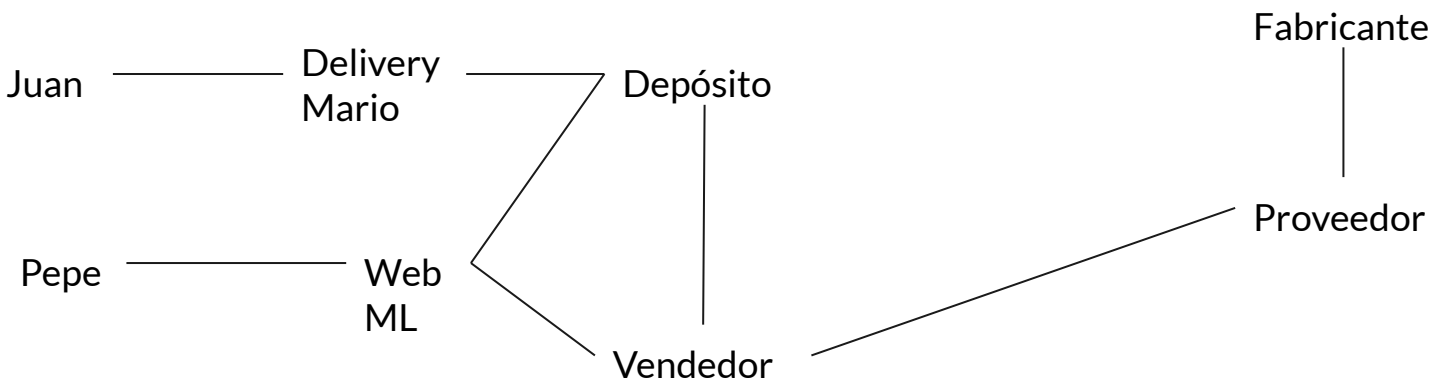
Descripción de Key

La programación orientada a objetos se basa en el principio del diseño recursivo:

1. Todo es un objeto.
2. Los objetos realizan cálculos haciéndose solicitudes entre sí a través del paso de mensajes.
3. Cada objeto tiene su propia memoria, que consiste en otros objetos.
4. Cada objeto es una instancia de una clase. Una clase agrupa objetos similares.
5. La clase es el repositorio del comportamiento asociado con un objeto.
6. Las clases se organizan en una estructura de árbol de una sola raíz, denominada jerarquía de herencia.

Ejemplo de los conceptos

Tomemos como ejemplo la compra de un artículo en Mercado Libre.

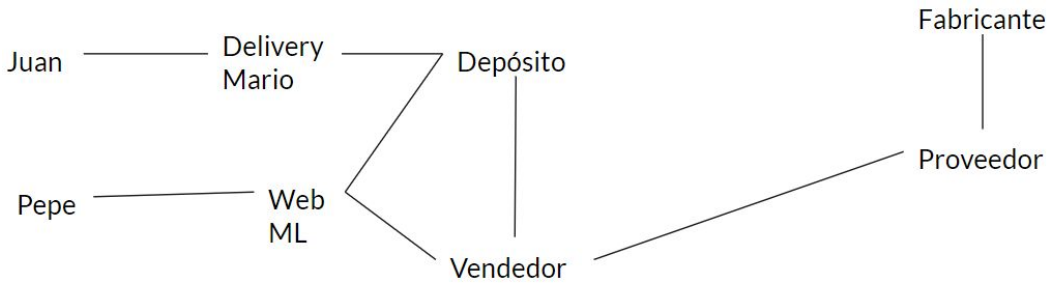


Agentes y comunidades

El resultado se logra a través de la interacción de agentes, los objetos.

Cualquier actividad no trivial requiere de la interacción de la comunidad entera de objetos trabajando en conjunto.

Cada objeto tiene un rol que cumplir, un servicio que proveer a la comunidad.



Elementos de POO - Objetos

Entonces tenemos el primer principio de Kay.

1. Todo es un objeto.

Las acciones en POO son realizadas por agentes, llamados instancias u objetos.

Hay muchos agentes en el ejemplo pepe, juan, delivery, depósito, etc. Cada agente tiene un papel que desempeñar, y el resultado se produce cuando todos trabajan juntos en la solución de un problema.

Elementos de POO - Mensajes

Y el principio número 2:

2. Los objetos realizan cálculos haciéndose solicitudes entre sí a través del paso de mensajes.

Las acciones en POO se producen en respuesta a solicitudes de acciones, llamadas mensajes. Una instancia puede aceptar un mensaje y, a cambio, realizará una acción y devolverá un valor.

Para comenzar el proceso de compra, Pepe le da un mensaje a la web de ML. Este, a su vez, le da un mensaje al vendedor y al depósito, y así sucesivamente.

Ocultamiento de la información

Tener en cuenta que cada objeto, como usuario de un servicio proporcionado por otro objeto, solo necesita saber el nombre de los mensajes que aceptará el objeto.

No necesita tener idea de cómo se llevarán a cabo las acciones realizadas en respuesta a la solicitud.

Habiendo aceptado un mensaje, un objeto se encarga de llevarlo a cabo.

Elementos de POO - Receptor

Los mensajes difieren de las llamadas a funciones tradicionales en dos aspectos muy importantes:

- En un mensaje hay un receptor designado que acepta el mensaje.
- La interpretación del mensaje puede ser diferente, dependiendo del receptor.

Aunque diferentes objetos pueden aceptar el mismo mensaje, las acciones (comportamiento) que realizará el objeto probablemente serán diferentes.

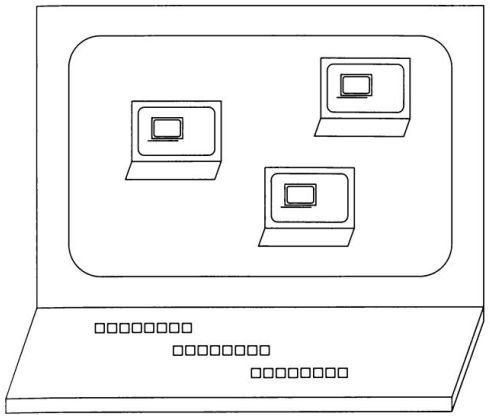
La determinación de qué comportamiento realizar se puede realizar en tiempo de ejecución, una forma de late binding.

El hecho de que el mismo nombre pueda significar dos operaciones completamente diferentes es una forma de polimorfismo (lo vemos en unidad 3).

Elementos de POO - Diseño Recursivo

3. Cada objeto tiene su propia memoria, que consiste en otros objetos.

Cada objeto es como una computadora en miniatura: un procesador especializado que realiza una tarea específica.



No interferencia

Es importante que se permita a los objetos realizar su tarea como mejor les parezca, sin interacciones innecesarias ni interferencias con otros objetos.

”Instead of a bit-grinding processor ... plundering data structures, we have a universe of well-behaved object that courteously ask each other to carry out their various desires” -- Dan Ingalls. (Desarrollador de Smalltalk)

“Ask not what you can do to your data structures, but ask what your data structures can do for you” -- Gerald Jay Sussman (autor de “Structure and Interpretation of Computer Programs”)

Elementos de POO - Clases

4. Cada objeto es una instancia de una clase. Una clase agrupa objetos similares.
5. La clase es el repositorio del comportamiento asociado con un objeto.

El comportamiento que se espera de Delivery Mario está determinado por una idea general del comportamiento de los deliverys.

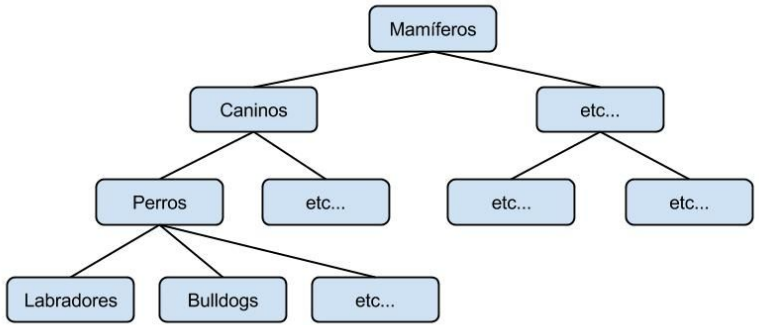
Decimos que Delivery Mario es una instancia de la clase Delivery.

El comportamiento está asociado con clases, no con instancias individuales. Todos los objetos que son instancias de una clase usan el mismo método en respuesta a mensajes similares.

Jerarquía de clases

Pero hay más que sabemos de Mario, además de que es delivery. Sabemos que es un un humano, un mamífero, un ser vivo, etc.

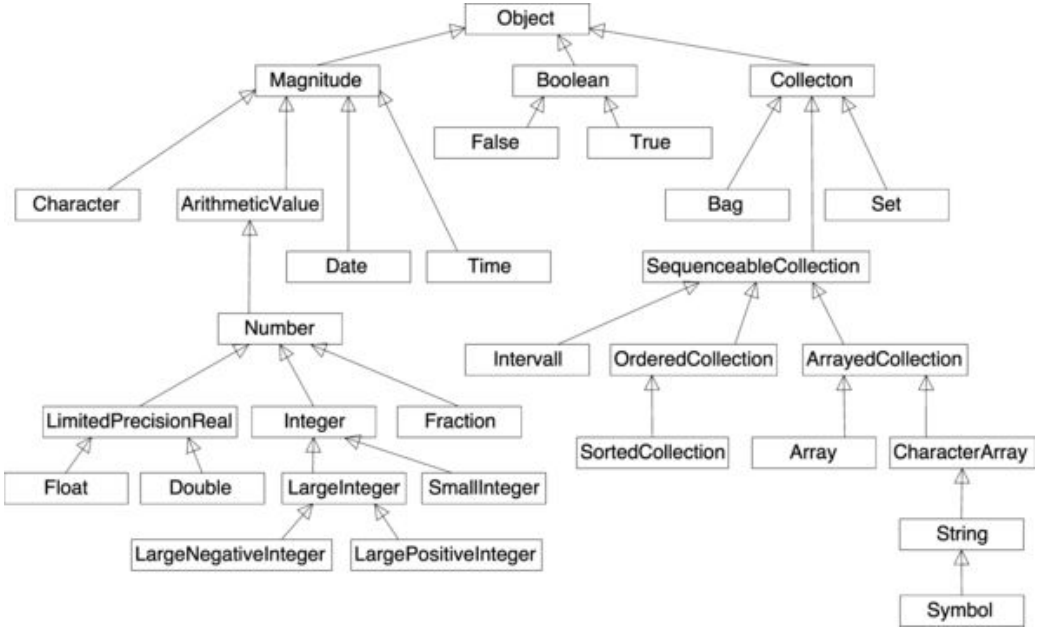
En cada nivel de abstracción tengo cierta información registrada. Esa información es aplicable a todos los niveles inferiores (más especializados).



Elementos de POO - Herencia

6. Las clases se organizan en una estructura de árbol de una sola raíz, llamada jerarquía de herencia.

La información (datos y/o comportamiento) que se asocia con un nivel de abstracción en una jerarquía de clases se aplica automáticamente a los niveles inferiores de la jerarquía.



Elementos de POO - Sobreescritura

Las subclases pueden alterar o anular la información heredada de las clases principales:

- Todos los mamíferos dan a luz crías vivas.
- Un ornitorrinco es un mamífero que pone huevos.

La herencia combinada con la anulación es donde se origina la mayor parte del poder de la Orientación a Objetos.

Metáfora y solución de problemas

Debido a que el punto de vista en POO es similar a la forma en que las personas resuelven problemas en la vida real (encontrar a otro agente para hacer el trabajo), la intuición, las ideas y la comprensión de la experiencia cotidiana pueden influir en la forma programar.

Resumen

- La programación orientada a objetos no es simplemente características añadidas a un lenguaje de programación. Más bien, es una nueva forma de pensar.
- La programación orientada a objetos ve un programa como una comunidad de agentes, denominados objetos. Cada objeto es responsable de una tarea específica.
- Un objeto es una encapsulación de estado (valores de datos) y comportamiento (operaciones).
- El comportamiento de los objetos está dictado por la clase de objeto.
- Un objeto exhibirá su comportamiento al invocar un método (similar a ejecutar un procedimiento) en respuesta a un mensaje.
- Los objetos y las clases amplían el concepto de tipos de datos abstractos al agregar la noción de herencia.

Bibliografía

Timothy A. Budd. An Introduction to Object-Oriented Programming. Pearson (2021)

Timothy Budd. Multiparadigm Programming in Leda. Addison-Wesley (1994) -->(leer capítulo 1)