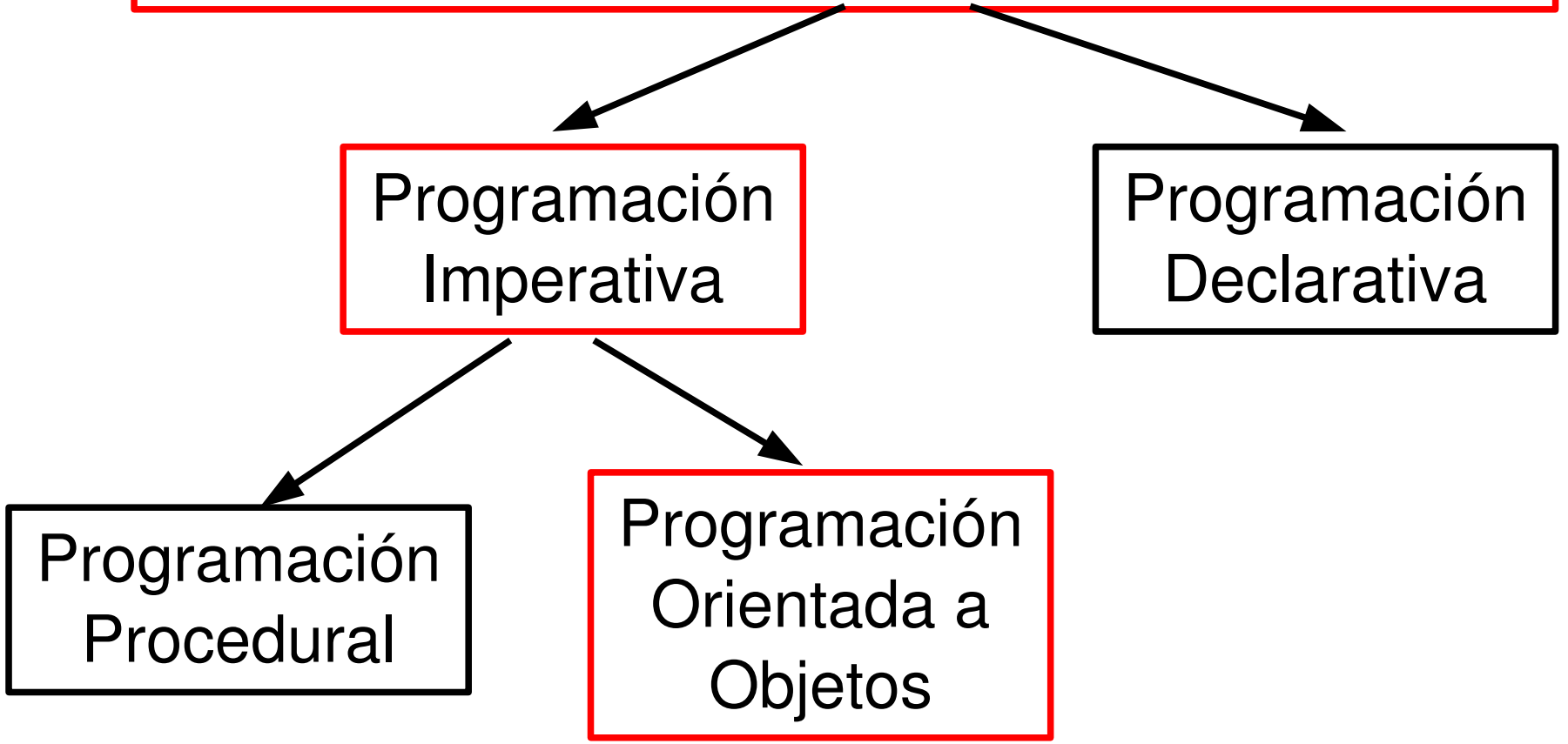
ELO329: Diseño y Programación Orientados a Objetos

Conceptos

Paradigmas de Programación



Paradigmas de Programación

- Historia:
  - Los computadores parten cableados por hardware,
  - Luego se introduce la programación en binario,
  - Se desarrolla el lenguaje assembler
  - Se desarrollan los lenguajes de alto nivel siguiendo dos paradigmas: Programación imperativa o declarativa
- Programación imperativa: donde la computación es descrita vía sentencias que cambian el estado del programa. Es una secuencia de comandos para el computador. ***El programa señala cómo se llega a la solución***. Ej. Python, C, Java, C++.
- Programación declarativa: la computación es descrita según su lógica sin indicar su control de flujo. ***Se indica qué debe hacerse no el cómo debe hacerse***. Ej. HTML, CSS, las fórmulas en planillas electrónicas.

Programación Imperativa

- Parte con la Programación por Procedimientos (Procedural Programming) donde ***la computación es descrita con el apoyo de llamados a procedimientos o funciones***. El programador debe encontrar la secuencia de instrucciones que resuelven la tarea, hace uso de procedimientos para mejorar la estructura y claridad del programa. Se dice que el lenguaje es estructurado (sin go-to).
- Luego evoluciona a la Programación Orientada a Objetos: El programador debe encontrar objetos; es decir, entidades que tienen comportamiento, estado y pueden interactuar con otros objetos. ***La computación se describe como la interacción de estos objetos***. Representa un intento por hacer los programas más cercanos a la forma como pensamos y nos relacionamos con el mundo. Este enfoque permite programas más naturales, más simples de construir bien y de entender.

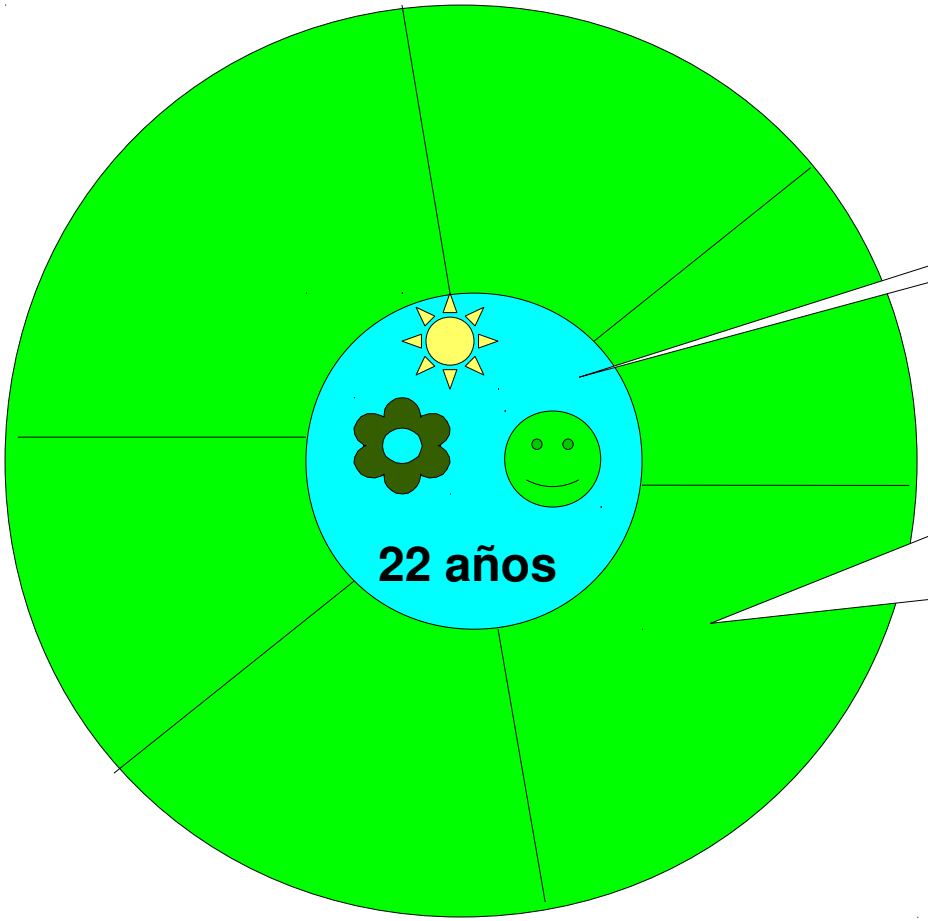
Programación Orientada a Objetos

- A partir de un problema el programador identifica los objetos del mundo real que intervienen en el problema.
- En el programa se crean objetos de software que modelan lo relevante de los objetos reales del problema. Además se crean objetos sintéticos (artificiales) que sean necesarios para estructurar una solución coherente y natural.

Objetos de Software

- Los objetos de software modelan dos aspectos de los objetos o entes reales: su estado y su comportamiento.
- Luego cada objeto de software tendrá un estado y cierto comportamiento.
- Además todo objeto de software tendrá un identificador o nombre para poder referirnos a él.
- Similar ocurre en C con:
  - integer i;
    - integer nos da una pista sobre qué cosas podemos hacer con i.
    - Si i=20, entonces podemos decir que su estado es 20.
    - i es un nombre necesario para diferenciarlo de otros enteros.

Objetos de software



Estado

Comportamiento, Servicios:
Cosas que el objeto puede hacer

Salvo excepciones, la interacción con el objeto sólo debería ser vía los servicios de cada objeto

Ejemplo de Objeto

- Un punto del espacio R^2
- Podemos representar un punto de varias formas: coordenadas cartesianas, polares, etc. Es así como podemos almacenar el estado de un punto como dos reales x e y, o dos reales r, ϴ.
- Independientemente de la forma como representemos un punto, nos puede interesar conocer:
  - El ángulo que forma el rayo del origen hasta el punto con el eje de abscisas.
  - Su distancia al origen.
  - Su distancia a otro punto, etc.

Un Punto en Java

- Una vez hecha la descripción para un punto, en java podríamos hacer cosas así:
- Punto p = new Punto();
- Con esto creamos un punto y tenemos un identificador o nombre p para referirnos a él. Su estado inicial es definido junto con su creación.
- Luego podríamos hacer cosas del tipo:
  - p.x() /* para obtener su coordenada x */
  - p.distancia() /* distancia al origen del punto p*/
  - p.distancia(p2) /* distancia entre p y otro punto p2 */

Clases

- Cada objeto es único, pero generalmente hay varios del mismo tipo. Hay varios puntos; por ejemplo.
- Cuando modelamos la realidad, lo hacemos reconociendo los objetos que comparten sus características. Por ejemplo: En un sistema podemos tener varios resortes, pero todos siguen el mismo patrón de comportamiento.
- Las clases definen las características de los objetos. Son el "rayado de la cancha" para una categoría de objetos de características comunes.
- Tendremos tantas clases como tipos de objetos distintos reconozcamos en nuestro problema.

Clases

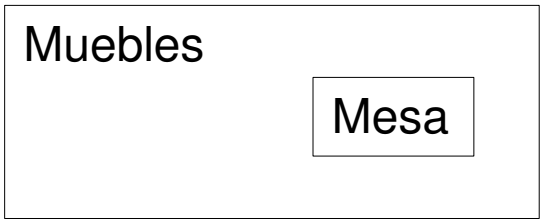
- Una clase debe definir todos los atributos (para almacenar el estado) y los comportamientos de ese tipo de objetos.
- El comportamiento (servicios o mensajes) que puede exhibir, ofrecer o recibir un objeto, lo expresamos como funciones en el sentido clásico de los lenguajes. Para diferenciarlos, en orientación a objetos se les llama métodos.
- Así, cada objeto posee atributos y métodos que son definidos en la clase a la cual él pertenece.

Ejemplo de Clase en Java

```
class Punto {  // nombre de la clase
  private int x,y;    // atributos para almacenar el estado
  public Punto(){    // método para definir estado inicial, al momento de ser creado
    x=y=0;              // este tipo de método lo llamamos constructor.
  }  // find e constructor
  public Punto(int _x, int _y){  // otro constructor
    x=_x;
    y=_y;
  }
  public int getX(){
    return x;
  }
  public int getY(){
    return y;
  }
  public boolean equals(Punto p){
    if (p== null) return false;
    return ((x==p.getX()) && (y==p.getY()));
  }
}
```

Jerarquías de clases

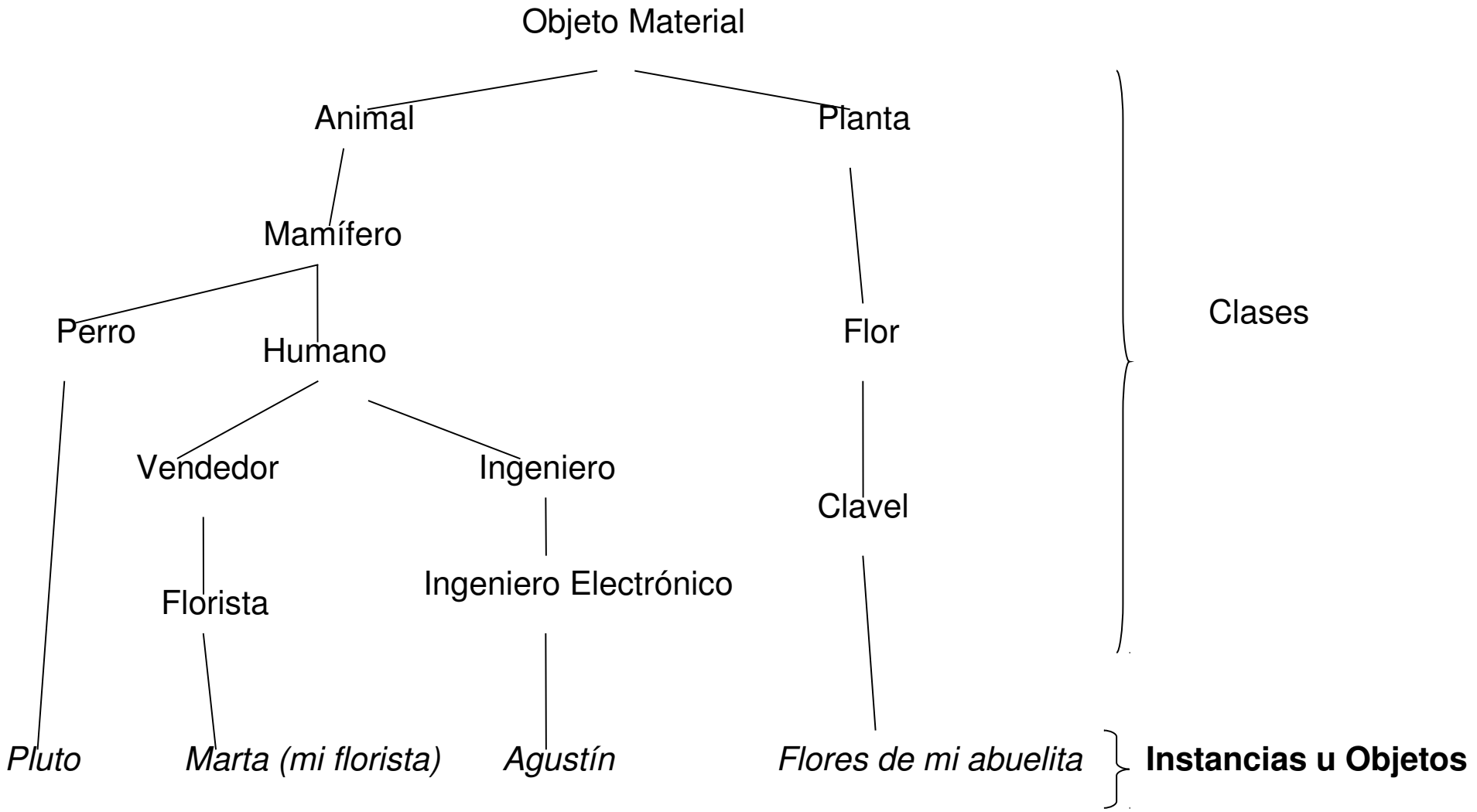
- Es común que los objetos del mundo real estén relacionados de la forma “es un”. Al ver la definición de casi cualquier cosa notamos:
  - Mesa: es un mueble que se compone de …
  - Chileno: es una persona natural de Chile....
- Es natural identificar jerarquías donde una clase comparte características comunes con otra clase y además posee algo propio que la distingue.



Jerarquías de Clases: Herencia

- Los Lenguajes Orientados a Objetos permiten definir clases a partir de clases ya definidas.
- El hecho que el conocimiento de una categoría más general es también aplicable a una categoría específica se conoce como Herencia.
- Decimos que la clase Mesa hereda los atributos de la clase Mueble, y ésta hereda de la clase Objeto_inanimado
- Según el problema .... Se establece una Jerarquía de clases.

Ejemplo: Jerarquías de Clases

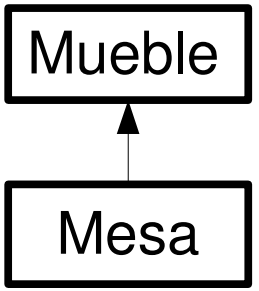


Vocabulario

- Así como Agustín es un caso específico de la clase Persona, podemos decir que Agustín es un ejemplo o instancia de persona.
- En OO decimos que los objetos son instancias de una clase. Al crear una instancia de un clase, creamos un objeto.
- Herencia: Manera de crear nuevas clases a partir de clases ya creadas. Así reusamos el trabajo hecho previamente.

Subtipos

- Cuando una clase hereda de otra, hablamos de clase base o padre y la otra es clases heredada o hija



Clase Base o Padre

Clase hija o heredada

- Es interesante ver que si en alguna situación requerimos un mueble, si tenemos una mesa estaríamos bien. Por ejemplo, si queremos bloquear una puerta, podemos usar un mueble; si tenemos una mesa cerca, ésta puede hacer el trabajo.
- Subtipo es el uso de un objeto en lugar de uno de jerarquía mayor. Mesa es subtipo de Mueble.

Subtipo

- Ejemplo: En la USM hay estudiantes, éstos son personas. Además hay estudiantes de Ing. Civil Electrónica, Telemática etc.
- Podemos identificar varias clases: Persona, Estudiante, EstudianteTelemática, EstudianteElectrónica.
- Los Lenguajes OO permiten que si en un método se debe poner una instancia de Persona como argumento, también es válido poner una instancia de Estudiante o una de EstudianteElectrónica.
- Esto es posible gracias a que los lenguajes OO permiten sustituir una instancia por otra proveniente de un subtipo.

Polimorfismo

- RAE: Cualidad de lo que tiene o puede tener distintas formas
- En OO esto ocurre de varias maneras.
- La idea básica es usar el mismo nombre para referirse a cosas similares. Supongamos la clase Stack: ¿Por qué debería darle un nombre distinto al método push cuando insertamos un real -float- o insertamos un carácter -char?
- Cuando un estudiante ocupa el lugar de una persona (por subtipo), también decimos que hay polimorfismo. El estudiante es también persona (dos formas).

Características de los POO

- Los lenguajes OO se caracterizan por:
  - Permiten expresar herencia: habilidad de reusar la definición de un tipo de objeto para definir otro tipo de objeto.
  - Subtipos: Si un objeto **a** tiene todo lo requerido por otro objeto **b**, entonces podemos usar **a** donde se esperaba **b**.
  - Permiten expresar abstracción: es decir, detalles de una implementación pueden ocultarse en el programa. Para usar una clase no necesitamos conocer cómo está implementada. La implementación de una clase es el código de sus métodos y los atributos que tiene.
  - Ligado dinámico: Cuando un método es invocado en un objeto, el código ejecutado (método) es determinado en tiempo de ejecución según el objeto que lo recibe. Esto conduce a que una misma invocación puede responder de manera distinta.

Diseño/Implementación Orientado a Objetos

- El Diseño OO involucra identificar los conceptos importantes de la solución y usar objetos para estructurar la manera cómo esos conceptos son reflejados en un sistema de software.
- Se trata de modelar el sistema como la interacción de objetos inter-actuantes.
- Involucra:
  - Identificar los objetos a un nivel de abstracción dado.
  - Identificar la semántica (comportamiento) de esos objetos.
  - Identificar la relación entre los objetos.
  - Implementar los objetos
- Es un proceso Iterativo