Introducción a la Programación Orientada a Objetos

Fundamentos de la Programación II

Índice

Lenguajes de programación.Definiciones.

- **Computación o cálculo**: aplicación de una secuencia de operaciones a un valor para obtener otro valor.
- **Programa**: especificación de una computación.
- **Lenguaje de programación**: notación para escribir programas.
- **Sintaxis** de un lenguaje de programación: estructura o forma de los programas.
- **Modelos de programación**: distintas formas de abordar un problema y darle una solución.
  - Ejemplos de modelos:
    - Imperativa.
    - Procedural o procedimental
    - Programación Orientada a Objetos.

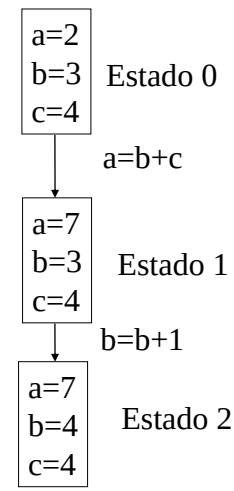
Programación Imperativa

Un programa es una secuencia de instrucciones que cambian el estado.

El código máquina está basado en el paradigma imperativo.

Programa Ejemplo

a=b+c
b=b+1



Programación Procedural o Procedimental

- El algortimo para abordar la solución se estructura en procedimientos que manipulan estructuras de datos y forman una jerarquía.

Programación Orientada a Objetos (POO)

- Un programa se construye a partir de componentes individuales llamados objetos.
- Un objeto encapsula datos y las operaciones que se pueden llevar a cabo sobre esos datos.
- En general, la programación se resuelve comunicando dichos objetos a través de mensajes (programación orientada a mensajes).
- Su principal ventaja es la reutilización de códigos y su facilidad para pensar soluciones a determinados problemas.

Conceptos de la OOP

- Clase
- Objeto
- Instancia
- Método
- Paso de Mensaje

Clase (I)

- Define la abstracción de las cosas (objetos), incluye sus **atributos** (características, campos) y sus **propiedades** (las cosas que puede hacer, o métodos, operaciones).
- Una clase es un “plano” o “molde” que describe la naturaleza de algo.
- Ejemplo: la clase **`Perro`** podría considerar la raza, color (**propiedades**), y la habilidades de ladrar y sentarse (**métodos**).
- Una clase describe el comportamiento de una familia de objetos.
  - Es una plantilla que se utiliza para definir los objetos
  - Puede verse como un tipo (que además define métodos)

Clase (y II)

- Las clases proporcionan modularidad y estructura en OOP.
- Una clase debería normalmente ser reconocida por una persona del dominio del problema que no sea programador.
- El significado de la clase debería tener sentido en el contexto al que se le da significado.
- El código de una clase debería ser relativamente autocontenido.
- Las propiedades y métodos definidos en una clase son sus **miembros**.

Ventaja de la utilización de clases

- Cada clase puede ser creada de modo independiente.
- Cada clase puede probarse de modo independiente.
- Asegura la consistencia de los datos pues ofrece un interfaz para su manejo.
- La implementación queda escondida al usuario de la clase (lo mismo que la implementación de los enteros queda oculta a los que los usan)
- Puede variarse la implementación sin tener que cambiar los programas que las utilizan.
- Es altamente reutilizable.

Objetos

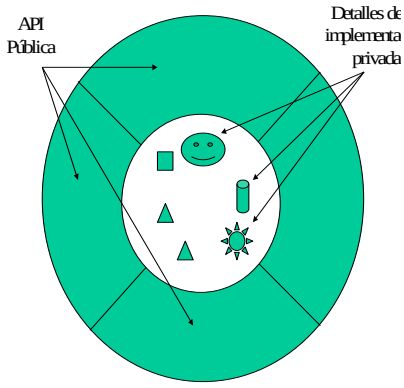
- Representan a los datos del problema real.
- Booch: “Entidad tangible que representa un comportamiento bien definido”
- Desde la perspectiva del conocimiento humano:
  - Algo tangible y/o visible.
  - Alguna cosa que pueda ser comprendida intelectualmente.
  - Algo hacia lo que se dirige el pensamiento o la acción.
- Representación de un elemento individual e identificable, real o abstracta, con un comportamiento bien definido en el dominio del problema.
- Un objeto es un ejemplar de una clase.

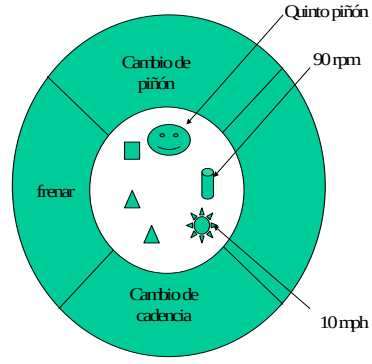
  Objeto = estado+comportamiento+identidad

Ejemplo

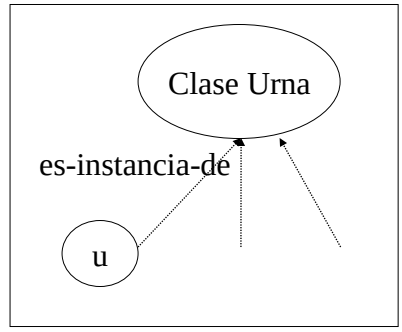
- La clase **`Perro`** define todos los posibles perros mediante un listado de las características y propiedades.
  - Un objeto **`Lassie`**, que es un perro particular, con versiones particulares de las características.
  - Un **`Perro`** tiene un color pelo. **`Lassie`** tiene el pelo de color marrón y blanco.

Una posible representación gráfica de objetos





Otra forma gráfica de representarlo



Objeto: Estado

- **Propiedades** o **atributos** que caracterizan al objeto.
- Cada atributo debe tener un **valor.**
- Los valores de los atributos pueden variar a lo largo de la vida del objeto
- Los atributos de un objeto son tenidos en cuenta según el dominio del problema
  - Si quiero vender un coche, los atributos interesantes son el precio, color, potencia, terminación, …
  - Si lo que quiero es participar en un rally, lo que interesa es aceleración, potencia, velocidad, anchura de ruedas.

Objeto: Comportamiento

- Viene determinado por la forma en la que el objeto interactúa con el sistema
- La forma de actuar sobre un objeto es enviándole un **mensaje**
- El mensaje activará un comportamiento del objeto (método) que será el que determine su forma de actuar
- Los métodos son los comportamientos del objeto
- Pueden generarse métodos para
  - Permitir consultas sobre aspectos internos del objeto.
  - Modificar atributos del objeto.
  - Envíe mensajes a otros objetos.
  - …

Objeto: Identidad

- Los objetos se identifican mediante propiedades características que los distinguen del resto de los objetos.
- Dos objetos con
  - Los mismos atributos
  - Los mismos valores en sus atributos
    - Son iguales pero no idénticos
- Para distinguir objetos se usan varias técnicas
  - Uso de direcciones de memoria o referencias
  - Uso de nombres definidos por el usuario
  - Uso de claves identificadoras internas o externas
- La identidad de un objeto permanece con él durante toda su vida.

Sobre el uso de objeto

- Implementación
  - Atributos: Se almacenan en una **memoria privada** que describe las propiedades del objeto.
  - Métodos o funciones miembro: Conjunto de operaciones que actúan sobre los atributos y definen su comportamiento.
- La forma en la que un método actúa sobre un objeto es a través del envío de **mensajes**. En un mensaje intervienen:
  - Receptor: Es el objeto que recibe el mensaje.
  - Selector: Es el objeto que envía el mensaje.

Instancia

- Una instancia es el objeto creado a partir de una clase en tiempo de ejecución.
- El objeto **`Lassie`** es una instancia de la clase **`Perro`**.
- El conjunto de valores de los atributos del objeto particular se conoce como **estado**.
- y el estado de un objeto se puede modificar mediante sus **métodos**.

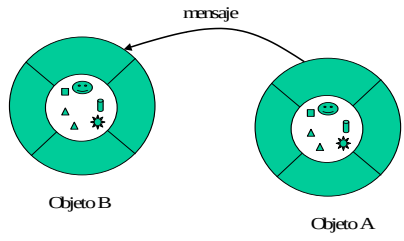
Método

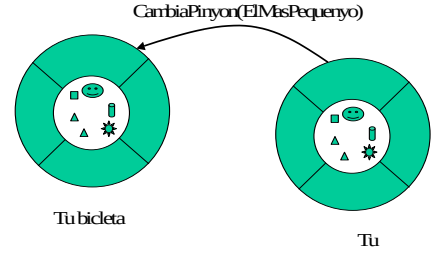
- Son las habilidades de un objeto.
- En un lenguaje, los métodos son verbos.
- **`Lassie`** es un **`Perro`**, que tiene la habilidad de ladrar. Por tanto, **`ladrar()`**, es un método de **`Lassie`**.
- Podría tener otros métodos también como, **`sentar()`**, **`comer()`**, **`caminar()`**, o **`correr()`**.
- Un método afecta solo a un objeto en particular. Todos los perros ladran, pero se necesita un solo perro concreto para que ladre.

Paso de Mensajes

- Es el proceso mediante el cual un objeto envía datos a otro objeto o pide a otro objeto que invoque a un método.
- Los mensajes que se pueden enviar a un objeto determinan su interfaz.
- Ejemplo, el objeto llamador **`Dueño`** podría decir al objeto **`Lassie`** que se siente mediante el paso del mensaje **`sentar`**, que invoca el método **`sentar`** de **`Lassie`**.

Ejemplo de paso de mensajes





Resultado del envío de un mensaje

- El resultado esperado del envío de un mensaje dependerá:
  - Del estado en que se encuentre dicho objeto
  - Del método involucrado en el mensaje
  - De la información que este mensaje pueda portar
- Un método tiene **total visibilidad** sobre los **atributos del objeto** al cual le han enviado el mensaje
  - Cuando a un objeto se le envía un mensaje, el método activado puede utilizar como variables los atributos del objeto receptor pudiendo incluso modificarlos
- El único camino para acceder al estado de un objeto **debe ser** por la activación de algún método, es decir, por el envío de un mensaje
- El conjunto de métodos que un objeto es capaz de responder es su **interfaz** y define su conducta.

Características de la OOP

- Abstracción
- Encapsulación
- Herencia
- Polimorfismo

Abstracción

- Según la RAE **abstraer**:

  Separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción
- Generalización de un determinado conjunto de objetos, de sus atributos y propiedades.
  - Sin preocuparnos de los detalles concretos de cada uno.
- Ejemplo:
  - Todos los perros son seres vivos que corren, ladran, muerden…, esto nos permite idenficar `Lassie` como un perro al verlo.

Ejemplo propuesto de Abstracción: Qué información puede ser relevante?

- Persona:
  - Atributos
    - DNI
    - Edad
    - Sexo
    - Altura
    - Peso
  - Métodos
    - Cambiar edad
    - Cambiar Peso
    - …
- Y para un coche? …

Encapsulación (I)

- Los datos y métodos de un objeto se encierran (encapsulan) en un objeto.
- Los datos sólo pueden ser modificados utilizando los métodos del objeto.
  - De esta forma se puede ocultar la representación interna de este objeto.
- El conjunto de métodos de un objeto que otros objetos pueden llamar se conoce como la **interfaz** de un objeto.
- Los demás objetos no conocen la implementación de los métodos de un objeto, sólo la forma de llamarlos.

Encapsulación (y II)

- La clase **`Perro`** tiene un método , **`ladrar()`**. El código del método **`ladrar()`** define exactamente como sucede (por ejemplo, **`inhala()`** y **`exhala()`**). **`Pedro`** el amigo de **`Lassie`**, no necesita saber como ladra.
- La encapsulación previene a los clientes de una interfaz depender de las partes de la implementación, facilitando los cambios futuros.

Encapsulación: Miembros público y privados de una clase

- Privados (`private`): son miembros que sólo pueden ser utilizados por miembros de esa clase.
  - Todos los atributos habitualmente deben ser privados.
- Públicos (`public`): son miembros que pueden ser utilizados por objetos de cualquier clase.
  - Determinan la interfaz de la clase.

ejemplo

- Pila de Enteros
  - públicos: los métodos `push/pop`.
  - privados: Tabla o lista que implementa la pila, y cima.

Ecuación fundamental de la POO

POO

=

Tipos abstractos de datos

+

Herencia

+

Polimorfismo

Tipos abstractos de datos

- Permiten
  - Encapsulación: guardar conjuntamente datos y operaciones que actúan sobre ellos
  - Ocultación: proteger los datos de manera que se asegura el uso de las funciones definidas para su manejo
- Ventajas:
  - Implementación escondida al cliente
  - Los tipos abstractos se generan independientemente
  - Los tipos abstractos se pueden probar independientemente
  - Aseguran la consistencia de los datos
  - Aumentan la reutilización de código

Herencia

•Incrementa la reutilización

- Maneja eficientemente relaciones "… es como un …"
- Crea nuevas clases a partir de generalizar o especializar otras clases ya existentes

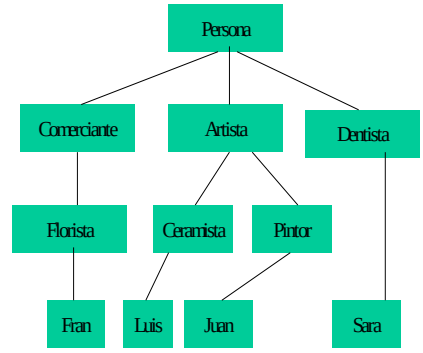
•Para ello a la hora de crear una clase puede reutilizar parte de la conducta de otra clase

- Esto se hace por medio de
  - Añadir o modificar métodos
  - Añadir variables de instancia

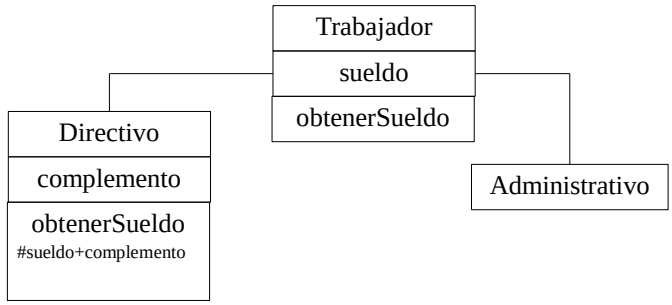
•Las 'Subclases' son versiones más especializadas de una clase, que **hereda** atributos y métodos de las clases **padres**, y pueden añadir las suyas propias.

Clase hija y clase padre (super)

- Si A hereda de B,
  - A es hija de B, A es subclase de B, A es derivada de B
  - B es padre de A, B es superclase de A, B es ancestro de A y de sus subclase, B es clase base de A
- La herencia puede ser
  - Simple: una clase hereda exclusivamente de otra clase
  - Múltiple: Una clase hereda de varias clases



Sobreescritura en la herencia



- obtenerSueldo de la clase Directivo sobreescribe el método obtenerSueldo de la clase Trabajador.
  - Son métodos diferentes.
- Si se llama a obtenerSueldo de un objeto Administrativo se ejecuta obtenerSueldo de Trabajador ya que no se sobreescribe.

Clases Abstractas

- Clases que proporcionan un interfaz común y básica a sus herederas.
- De ella no se obtendrá ninguna instancia.
- Definirá métodos con el cuerpo vacío o métodos con comportamientos generales a todas las subclases.

Polimorfismo

- Es la habilidad de los objetos de responder con distintos comportamientos ante un mismo mensaje.
- En OOP
  - Misma interfaz de método para distintos comportamientos (mismo nombre del método con distintas declaraciones y comportamientos).
  - Sobrecarga de métodos.

Ejemplo de polimorfismo

- Clase ListaDeLaCompra

  añadir(producto)

  añadir(otraListaDeLaCompra)
- Son métodos diferentes.
- Dependiendo del parámetro el comportamiento es uno u otro

Descrípción de una clase

```
Clase Urna
 Variables de Instancia privadas
   Número de bolas blancas (blancas)
   Número de bolas negras (negras)
 Métodos públicos
   sacaBola()
   meteBola(Color)
   quedanBolas()
   quedaMasDeUnaBola()
 Métodos privados
   totalBolas()
Fin Clase
```

Referencias al propio objeto

Un objeto puede enviarse un mensaje a sí mismo

```
int quedaMasDeUnaBola(){
  return (1 < mimismo.totalBolas());
}
```

- **mimismo** = self, current, this
- Hay lenguajes que la referencia a "**mimismo**" puede suprimirse

```
int quedaMasDeUnaBola(){
  return (1 < totalBolas());
}
```

Programadores en la POO

- Productores de clases
- Consumidores de clases
  - Un programa puede contener instancias de clases previamente definidas y clases definidas específicamente para este programa.
  - La relación entre variables de instancias y los objetos de una clase se define como una relación de tipo “es parte de…”
    - Así, el atributo **número de bolas blancas** de la urna **u** es parte de la urna u
  - Un objeto puede contener como una parte suya a otros objetos. **Agregación**
  - Un objeto puede contener como una parte suya referencias a otros objetos. **Asociación**
  - **Composición: Agregación o asociación**

Dependencia del lenguaje

- Las posibilidades de ocultación de la información son dependientes del lenguaje
  - Hay lenguajes que ponen diferentes niveles de privacidad
  - Hay lenguajes que no pueden hacer las variables de instancia privadas

Una urna heredada

- Sea **UrnaTrampa** una clase que hereda las propiedades de la clase **Urna**.
- La clase **UrnaTrampa** es como una **Urna** pero con un comportamiento especial

```
Clase  UrnaTrampa hereda Urna
   Variables de Instancia privadas
      contExtracciones
   Métodos Privados
      cambiaBolas()
Fin Clase
```

  - No se indican los métodos que hereda de la clase **Urna**
  - Una vez definida, se usa como una clase más
  - Una clase puede redefinir los métodos de la clase de la que hereda. La nueva definición puede apoyarse en la antigua.