ADSI

Introducción a POO

Instructor: Gustavo Adolfo Rodríguez Q.

garodriguez335@misena.edu.co

ADSI

1. INTRODUCCION A LA PROGRAMACION ORIENTADA A OBJETOS

Dentro de la ingeniería del desarrollo de software, existen diferentes tipos de enfoques particulares o filosofías que se utilizan para ayudar a dar solución a un problema determinado. Estos enfoques o filosofías junto con sus técnicas reciben el nombre de **paradigmas de programación.**

Existe una gran variedad de paradigmas de programación, donde cada uno de ellos tiene sus ventajas y desventajas. A continuación se listan algunos de los paradigmas de programación más reconocidos y utilizados:

- Paradigma por procedimientos
- Paradigma estructural
- Paradigma funcional
- Paradigma orientado a objetos
- Paradigma orientado a aspectos

No existe un paradigma superior o mejor que otro, sólo existen paradigmas más adecuados para el tipo de desarrollo requerido. Así por ejemplo, en el paradigma por procedimientos lo más importante es el procedimiento (implementación del algoritmo) que se desarrolla utilizando un lenguaje de programación cualquiera.

Los programas son simulaciones de modelos conceptuales y físicos. Estos modelos son, muy a menudo, complejos por naturaleza y es tarea del programador reducir esa complejidad a algo comprensible por el usuario. Por tanto, las interfaces de usuario modernas extraen muchas herramientas a partir de unos cuantos conceptos familiares, como el menú desplegable, el escritorio, la papelera… maximizando nuestra limitada capacidad de gestionar tareas complejas. Del mismo modo que es importante organizar la interfaz de usuario para que todas sus opciones sean accesibles de forma lógica, también es importante una organización en lo que a la programación se refiere.

Cuando un programador crea un programa de software, el número de variables e interacciones que se tienen que tratar de forma simultánea se convierten en una carga muy pesada cuando el programa crece.

Este problema es mayor en los entornos de programación en los que se piensa que los programas son una serie de pasos lineales, como se considera en el *paradigma por procedimientos*. Los lenguajes basados en procedimientos, como C, han empleado este modelo con un gran éxito hasta hace unos años. Este modelo es adecuado para programas simples pero presenta problemas cuando los programas se hacen muy grandes. La ausencia de interfaces definidas o interacciones entre los elementos del programa, junto con el número total de éstas, hace que ya sea imposible su manejo y tenga una tasa de probabilidad de errores demasiado elevada.

Para limitar el número de cosas simultáneas que los programadores tienen que tratar o gestionar de forma efectiva en su propio código nace el nuevo enfoque, el ***paradigma de***

programación orientada a objetos. Los humanos gestionamos la complejidad a través de la abstracción. Por ejemplo, nosotros no vemos a nuestra casa como un conjunto de cosas indescifrables; la vemos como un objeto bien definido con un comportamiento único y predecible. Esta abstracción nos permite saber que en nuestra casa podemos vivir, dormir, compartir en familia sin que nos abrume la complejidad que representa una casa y todos sus sistemas.

La abstracción nos permite ignorar de momento todos los sistemas que hacen habitable nuestra casa, así desde el exterior una casa es simplemente eso: una casa, pero en realidad una casa está compuesta por varios sistema internos tales como los muros, los cimientos, las tuberías, los drenajes, el cableado y más.

De la misma forma en que hemos descompuesto nuestra casa en un conjunto de sistemas definidos que trabajan juntos para hacer habitable nuestra casa, los programas software también se pueden descomponer en varios objetos. Una secuencia de pasos de un proceso se puede convertir en un intercambio de mensajes entre objetos autónomos, cada uno con su comportamiento único. Tratamos estos objetos como entidades del mundo real que responden a mensajes que les dicen que hagan algo. Es la esencia de la programación orientada a objetos.

Nuestros objetos están dotados de personalidad y confiamos en que siempre se comportarán de acuerdo a las especificaciones que están definidas para ellos. Sabemos que al activar el interruptor de la luz de la sala de nuestra casa esa luz se encenderá sin causar que un muro se caiga o que los drenajes se desborden. Podemos confiar en lo anterior puesto que los sistemas de nuestra casa han sido tratados como objetos con características especiales y comportamiento bien definido ante estímulos llamados ***mensajes.***

Si miramos a nuestro alrededor vemos que todo lo tangible y lo intangible puede ser descrito o modelado utilizando el paradigma de orientación a objetos. Para utilizar el paradigma de orientación a objetos de forma adecuada es necesario hacer algunas precisiones y definiciones.

1.1. Objetos

Casi todo puede ser considerado un objeto. El dinero, un helicóptero, una bicicleta, los perros, un carro. Los objetos representan cosas, simples o complejas, reales o imaginarias. Una antena parabólica es un objeto muy complejo pero real. Un objeto profesor, representa los detalles y actividades de una persona, no es esa persona en sí misma, es pues, imaginario. Una frase, un número complejo, una receta y una cuenta bancaria también son representaciones de cosas intangibles. Todas ellas son objetos.

Algunos objetos pueden ser divididos a su vez en otros objetos que los componen por ejemplo, un carro es un objeto complejo que está compuesto de otros objetos más simples tales como llantas, timón, motor, sillas etc.

Algunas cosas no son objetos, sino ***atributos***, valores o características de objetos. Es decir, no todas las cosas son objetos, ni son consideradas normalmente como objetos. Algunas de ellas son simplemente atributos de los objetos como el color, el tamaño y la velocidad. Los atributos reflejan el estado de un objeto, la velocidad del objeto *carro*, o el tamaño de un objeto *llanta*. Normalmente no tiene sentido considerar la *velocidad* o el *tamaño* como un objeto.

Además de definir los atributos de un objeto, también podemos definir las acciones o comportamientos de éste. Podemos decir que un objeto *carro* puede acelerar, frenar, transportar. Cada una de estas acciones se denomina ***método.***

En conclusión, si estamos utilizando el paradigma de orientación a objetos y deseamos definir un objeto particular debemos especificar de ese objeto la siguiente información fundamental:

- **Nombre del objeto :** Nombre que identifica al objeto
- **Abstracción de datos:** Un listado de los atributos que tiene el objeto. También es necesario definir el valor que tiene dicho atributo.
- **Abstracción funcional:** Un listada de los métodos del objeto.

A continuación se desarrolla la definición de un objeto, utilizando el paradigma de orientación a objetos.

NOMBRE	pepipo_perez
ABSTRACCION DE DATOS	**Nombre:** Pedro
	Apellido: Perez
	Edad: 45
	Estatura: 180 cm
	Nacionalidad: Colombiano
	Color de piel: Blanco
	Color de ojos: Negro
	Contextura: Delgado
	Mayor de edad: Si
ABSTRACCION FUNCIONAL	Hablar()
	Caminar()
	Correr()
	Reir()
	Llorar()

Si prestamos atención a la lista de atributos que componen un objeto podemos darnos cuenta que todos los atributos son **adjetivos**, mientras que los métodos son **verbos.**

En el ejemplo anterior, el objeto descrito se considera un **objeto atómico** puesto que todos sus atributos son de tipos de datos básicos (Enteros, Cadenas, Boleanos, etc.), pero existen algunos objetos que dentro de sus atributos incorporan a otros objetos. Los objetos que incorporan a otros objetos dentro de sus atributos se denominan **objetos complejos.**

A continuación se desarrolla la definición de un objeto complejo, utilizando el paradigma de orientación a objetos.

NOMBRE	micarro
ABSTRACCION DE DATOS	**Color:** Verde
	Tamaño: Grande
	Velocidad: 200 Km/h
	Aceleracion: 1.5m/s²
	Motor: Motor
	Freno: Freno de Mano, Freno de Pedal
	Marca: Toyota
	Modelo: Prado
	Año: 2010
	Matricula: BCL 173
	Averiado: No
ABSTRACCION FUNCIONAL	Acelerar()
	Frenar()
	Ancender()
	Apagar
	Transportar()

Podemos apreciar que el anterior ejemplo incorpora dentro de sus atributos un objeto de tipo Motor llamado motor. También podemos apreciar que el objeto incorpora dos atributos llamados Mano, pedal que son de tipo Freno, donde Freno y Motor son Objetos atómicos.

1.2. Clases

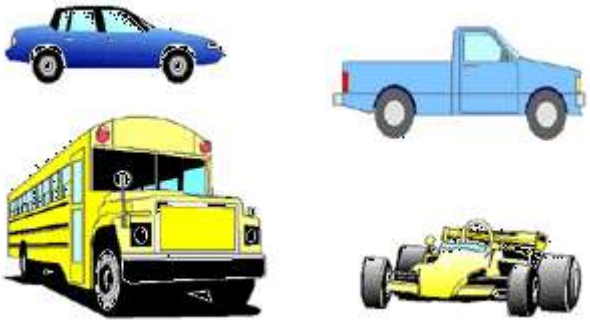
Como se ha dicho antes, todas las cosas del mundo sean tangibles o intangibles pueden ser modeladas como objetos, así, existen un sinnúmero de objetos los cuales pueden ser agrupados.

Una agrupación de objetos se denomina ***clase*** la palabra ***tipo*** también es utilizada como sinónimo de clase.

La condición necesaria para agrupar los objetos en clases es que todos ellos comparten los mismos atributos en cuanto a número y tipo, y también comparten el comportamiento (métodos).

Si decimos que una clase es una agrupación de objetos que comparten atributos y métodos comunes, también es válido decir que un objeto es una ***instancia*** particular de una clase.

En el siguiente caso, cada uno de los dibujos de la figura son ***instancias*** de la clase **Carro**



La clasificación de los objetos puede ser arbitraria. Por ejemplo, una clase **Transporte** puede incluir objetos *tren*, objetos *carro* o incluso, objetos *caballo*. El hecho de que esos objetos **Transporte** funcionen de forma diferente es irrelevante para el problema que se intenta resolver. Si **Transporte** sólo se preocupa del traslado de un sitio a otro, la implementación subyacente es irrelevante.

Si estamos utilizando el paradigma de orientación a objetos y deseamos definir una clase cualquiera debemos especificar de esa clase la siguiente información fundamental:

- **Nombre de la clase:** Nombre que identifica a la clase
- **Atributos de clase:** Un listado de los atributos que tendrán todos los objetos de esta clase. También es necesario especificar el tipo de dato del atributo.
- **Métodos de clase:** Un listada de los métodos que tendrán los objetos de esta clase.

A continuación se desarrolla la definición de la clase Persona, utilizando el paradigma de orientación a objetos.

NOMBRE DE CLASE	Persona
ATRIBUTOS DE CLASE	Cadena de caracteres : **nombre**
	Cadena de caracteres : **apellido**
	Entero : **edad**
	Entero : **estatura**
	Cadena de caracteres : **nacionalidad**
	Cadena de caracteres : **color de piel**
	Cadena de caracteres : **color de ojos**
	Cadena de caracteres : **contextura**
	Boleano: **mayor de edad**
METODOS DE CLASE	Hablar()
	Caminar()
	Correr()
	Reir()
	Llorar()

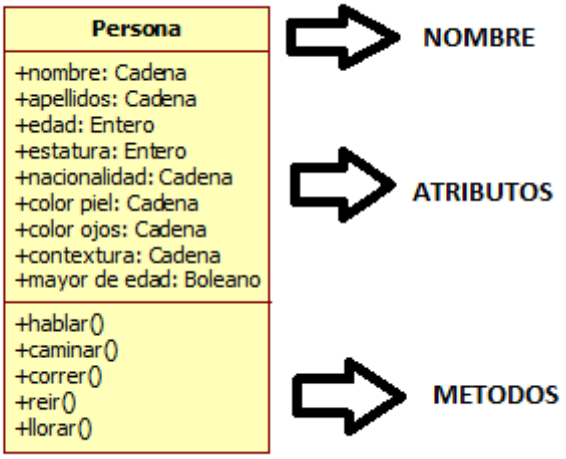
Como se puede apreciar, la definición de clase es la plantilla que se utiliza para la creación de los objetos. Si analizamos con detenimiento podemos darnos cuenta que esta definición de clase fue la que se utilizó para crear al objeto ***pepito_perez.*** Por tanto, podemos decir que el objeto *pepito_perez,* es una ***instancia*** de la clase *Persona.*

En la definición de clase se describen los atributos de tendrán los objetos, mientras que cuando se crea un objeto instancia de una clase, se le da valores a esos atributos.

1.3. Notación UML

El lenguaje unificado de modelado (UML por la sigla en Inglés) define la forma de describir clase de forma gráfica.

A continuación se muestra en una figura la representación en UML de la clase Persona de la sección anterior.



1.4. Interacciones entre objetos

El comportamiento de un sistema depende de las interacciones que tengan los objetos que lo componen. Por ejemplo:

- El objeto acelerador interactúa con el objeto motor en un carro.
- El objeto llave interactúa con el objeto puerta en una casa.

El modelado de objetos no sólo modela los objetos en un sistema, sino también sus interrelaciones. Para realizar su tarea, un objeto puede delegar trabajos en otro. Este otro puede ser parte integrante de él o ser cualquier objeto del sistema. Pero no todos los objetos pueden comunicarse con todos, existe unas formas establecidas y normas para el intercambio de mensajes. Esas normas están determinadas por el ***tipo de acceso*** que tenga un objeto, un método o un atributo.

1.5. Tipos de Acceso

1.5.1.Acceso Público

Este tipo de acceso hace referencia a que todos los objetos tanto de la misma clase como de clases diferentes pueden enviar mensajes o alterar el estado.

1.5.2.Acceso Protegido

Este tipo de acceso hace referencia a que sólo los objetos de la misma clase pueden enviar mensajes o alterar el estado.

1.5.3.Acceso Privado

Este tipo de acceso hace referencia a que sólo el objeto propietario puede enviar mensajes o alterar el estado.

Los tipos de acceso son más fácilmente comprensibles cuando se utiliza un lenguaje de programación orientado a objetos. Estos aspectos los veremos en unidades posteriores.

1.6. Herencia entre clases

Cuando en el mundo real hablamos de *herencia*, relacionamos el concepto con algo que un sujeto de nivel superior (padre) de cede a otro sujeto (hijo).

En el paradigma de programación orientada a objetos también se utiliza la herencia y tiene un concepto un poco parecido al que manejamos en la vida real.

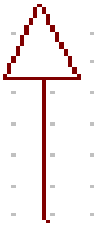
En programación orientada a objetos se utiliza la herencia cuando queremos hacer nuevas clases más especializadas o específicas a partir de una clase base denominada ***clase padre.*** La nueva clase más especializada generalmente se denomina ***clase hija.*** La clase hija hereda todos los atributos y métodos siempre y cuando estos no hayan sido declarados con tipo de acceso privado por parte de la clase padre.

Las clases hijas pueden definir sus propios atributos y métodos, e incluso pueden sobrescribir el comportamiento de un método heredado de la clase padre.

Supongamos que tenemos una clase llamada *Avión,* esta clase representa a todos los aviones que existen en el mundo, pero se requiere tener mayor especificidad en cuanto al tipo de avión puesto que existen aviones de carga, de pasajeros, privados, militares, de fumigación. Cada uno de esos tipos de aviones comparten características comunes pero también tienen atributos que sólo tienen sentido para un tipo particular. Por ejemplo, para un avión de pasajeros es importante el número azafatas necesarias para atender a los pasajeros, mientras que este atributo carece de sentido en un avión de fumigación o un avión militar. Para un avión militar es importante la cantidad de misiles que puede disparar a un oponente pero este atributo carece de sentido para un avión de pasajeros o un avión privado.

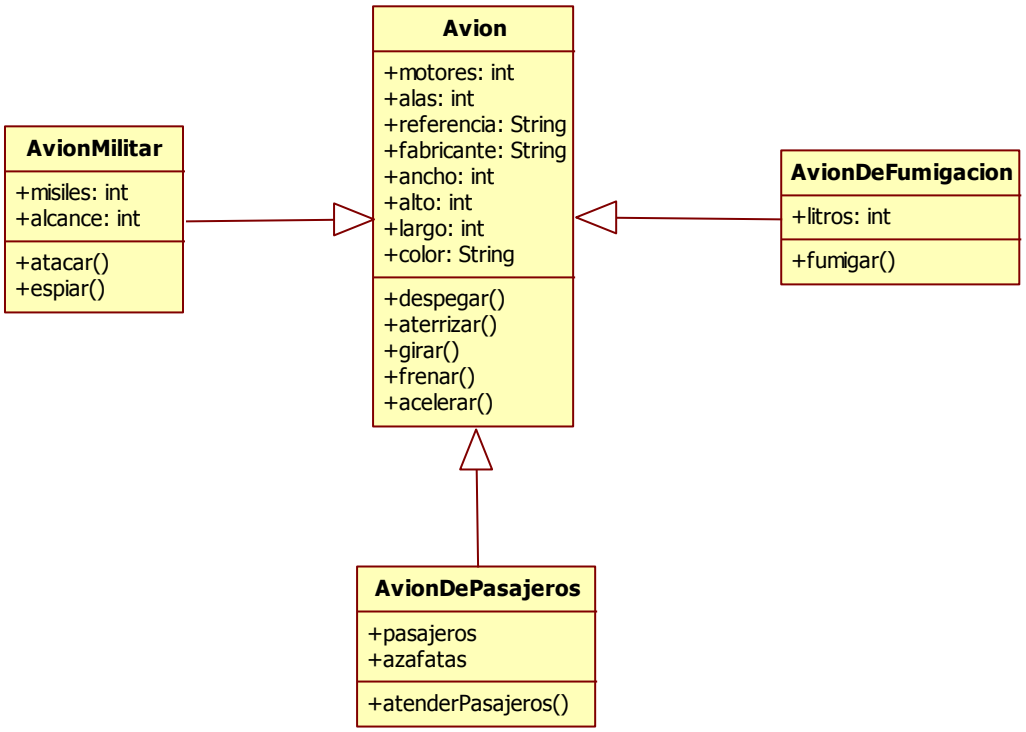
Si queremos tener bien definidas a las clases AvionMilitar, AvionPrivado, AvionDePasajeros, AvionDeCarga y AvionDeFumigacion debemos entonces hacer que todas ellas hereden de la clase Avion y además agregar los atributos y métodos propios y pertinentes para cada clase hija.

La notación UML define que la herencia entre clases de denota con una flecha como la que se muestra a continuación



La punta de la flecha identifica a la clase padre mientras que la cola de la flecha identifica a la clase hija.

El modelo UML que describe la herencia que existe entre la clase Avion y las clases AvionMilitar, AvionDePasajeros y AvionDeFumigacion se muestra a continuación:



Puesto que la clase AvionMilitar hereda de la clase Avion, cada objeto instancia de la clase AvionMilitar tendrá los siguientes atributos:

motores, alas, referencia, fabricante, ancho, alto, largo, color, misiles y alcance.

Los objetos de la clase AvionMilitar tendrán los siguientes métodos:

Despegar, aterrizar, girar, frenar, acelerar, atacar y espiar.