SECRETARIA DE EDUCACIÓN PÚBLICA

DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA

INSTITUTO TECNOLÓGICO DE VERACRUZ

SIMULACION 1

UNIDAD 3 HERRAMIENTA DE PROGRAMACION DE MODELOS DE SIMULACION NUMERICA

3.4 PROGRAMACION ORIENTADA A OBJETO POO

CATEDRÁTICO:
DR. JOSE ANTONIO GARRIDO NATAREN

FECHA DE ENTREGA:
Veracruz Ver a 19 de Marzo del 2015

INTRODUCCIÓN

Los objetos son entidades que tienen un determinado *estado*, *comportamiento (método)* e *identidad*:

- El *estado* está compuesto de datos o informaciones; serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El *comportamiento* está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La *identidad* es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Los **métodos (comportamiento)** y **atributos (estado)** están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una **programación estructurada camuflada** en un lenguaje de programación orientado a objetos.

La POO difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada solo se escriben funciones que procesan datos. Los programadores que emplean programación orientada a objetos, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

3.3 PROGRAMACION ORIENTADA A OBJETO

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.
Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación.

La POO representa una metodología de programación que se basa en las siguientes características:

1) Los diseñadores definen nuevas clases (o tipos) de objetos.
2) Los objetos poseen una serie de operaciones asociadas a ellos.
3) Las operaciones tienden a ser genéricas, es decir, operan sobre múltiples tipos de datos.
4) Las clases o tipos de objetos comparten componentes comunes mediante mecanismos de herencia.

Motivación

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Que es una serie de normas de realizar las cosas de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.

La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador. Aunque podamos hacer los programas de formas distintas, no todas ellas son correctas, lo difícil no es programar orientado a objetos sino programar bien.
Programar bien es importante porque así nos podemos aprovechar de todas las ventajas de la POO.

Elementos y Características de la POO

Los elementos de la POO, pueden entenderse como los materiales que necesitamos para diseñar y programar un sistema, mientras que las características, podrían asumirse como las herramientas de las cuáles disponemos para construir el sistema con esos materiales.

Entre los elementos principales de la POO, podremos encontrar a:

Clases en POO

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo

que estamos haciendo es programar una clase. En los ejemplos anteriores en realidad hablábamos de las clases coche o fracción porque sólo estuvimos definiendo, aunque por encima, sus formas.

Una clase se divide en una parte pública y en una parte privada. El nombre de la clase debe ser único.

Parte pública

Describe a qué operaciones responden los objetos de una clase (cómo se comportan los objetos). En esta parte de la clase se declaran las cabeceras de los métodos de la clase que podrán ser "invocados" por los objetos. Es decir, si un método se declara en la parte pública, podrá ser "invocado" por un objeto de dicha clase, de lo contrario no podrá ser "invocado" por un objeto. Es la parte "visible" de la clase, la interfaz de la clase.

Parte privada

Describe los datos de la clase y cómo las operaciones manipulan dichos datos. Esta parte de la clase es donde se oculta (encapsula) la información de la clase: datos e implementación de métodos declarados o no en la parte pública de la clase. Es una parte "no visible", cada objeto de una determinada clase tiene sus atributos (datos) y sus métodos.

Propiedades en clases

Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

Métodos en las clases

Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

Objetos en POO

Es la entidad en torno a la cual gira la POO. Un objeto es un ejemplar concreto de una clase, como por ejemplo el curso de metodología de la programación es un curso concreto dentro de todos los tipos de cursos que pueden existir. Un objeto pertenece a una clase, por lo tanto dispondrá de los atributos (datos) y operaciones (métodos) de la clase a la que pertenece. Un objeto responde al comportamiento definido por las operaciones de la clase a la que pertenece.Para crear un objeto se tiene que escribir una instrucción especial que puede ser distinta dependiendo el lenguaje de programación que se emplee, pero será algo parecido a esto.

```
miCoche = new Coche()
```

Con la palabra new especificamos que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis podríamos colocar parámetros con los que inicializar el objeto de la clase coche.

En la POO, un objeto no es algo "eterno", se **instancian** (crean) y se **destruyen**. Una vez que se haya instanciado un objeto puede **recibir mensajes**. Los objetos pueden instanciarse de forma estática o de forma **dinámica** (recordar estos conceptos de capítulos anteriores). Un objeto estático comienza su existencia una vez es declarado (instanciación de objetos estáticos), sin embargo, un objeto dinámico no comienza su existencia al ser declarado, sino al recibir un espacio de memoria (instanciación dinámica de objetos).

Estados en objetos

Cuando tenemos un objeto sus propiedades toman valores. Por ejemplo, cuando tenemos un coche la propiedad color tomará un valor en concreto, como por ejemplo rojo o gris metalizado. El valor concreto de una propiedad de un objeto se llama estado.
Para acceder a un estado de un objeto para ver su valor o cambiarlo se utiliza el operador punto.

```
miCoche.color = rojo
```

El objeto es miCoche, luego colocamos el operador punto y por último el nombre de la propiedad a la que deseamos acceder. En este ejemplo estamos cambiando el valor del estado de la propiedad del objeto a rojo con una simple asignación.

Los objetos "se declararán" como atributos de las clases, en las cláusulas van de los métodos o en la lista de parámetros de los métodos. El único objeto que debe ser "declarado" en el programa principal será el de la clase raíz de toda la jerarquía de clases (por cuestiones del lenguaje Pascal orientado a objetos). Los objetos estáticos existen durante la ejecución del programa. Los objetos dinámicos existen mientras no sean destruidos (liberación de la memoria del objeto), mediante la sentencia dispone.

Mensajes en objetos

Un mensaje en un objeto es la acción de efectuar una llamada a un método. Por ejemplo, cuando le decimos a un objeto coche que se ponga en marcha estamos pasándole el mensaje ponte en marcha.

Para mandar mensajes a los objetos utilizamos el operador punto, seguido del método que deseamos invocar.

```
miCoche.ponerseEnMarcha()
```

En este ejemplo pasamos el mensaje ponerseEnMarcha(). Hay que colocar paréntesis igual que cualquier llamada a una función, dentro irían los parámetros.

Tras ser instanciado, un objeto consta de los atributos de la clase a la que pertenece, y puede recibir mensajes. Dicho objeto determinará a qué método de su clase corresponde dicho mensaje. Un objeto no podrá recibir mensajes correspondientes a métodos de una clase a la que no pertenezca.

Otras cosas

Hay mucho todavía que conocer de la POO ya que sólo hemos hecho referencia a las cosas más básicas. También existen mecanismos como la herencia y el polimorfismo que son unas de las posibilidades más potentes de la POO.

La **herencia** sirve para crear objetos que incorporen propiedades y métodos de otros objetos.

El **polimorfismo** sirve para que no tengamos que preocuparnos sobre lo que estamos trabajando, y abstraernos para definir un código que sea compatible con objetos de varios tipos.

Ventajas de la Programación Orientada a Objetos

Vamos a ver las ventajas más importantes de la programación orientada a objetos:

- ***Reusabilidad*.** Cuando hemos diseñado adecuadamente las clases, se pueden usar en distintas partes del programa y en numerosos proyectos.
- ***Mantenibilidad*.** Debido a la sencillez para abstraer el problema, los programas orientados a objetos son más sencillos de leer y comprender, pues nos permiten ocultar detalles de implementación dejando visibles sólo aquellos detalles más relevantes.
- ***Modificabilidad*.** La facilidad de añadir, suprimir o modificar nuevos objetos nos permite hacer modificaciones de una forma muy sencilla.
- ***Fiabilidad*.** Al dividir el problema en partes más pequeñas podemos probarlas de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

Lenguajes Orientados a Objetos

Simula (1967) es aceptado como el primer lenguaje que posee las características principales de un lenguaje orientado a objetos. Fue creado para hacer programas de simulación, en donde los "objetos" son la representación de la información más importante. Smalltalk (1972 a 1980) es posiblemente el ejemplo canónico, y con el que gran parte de la teoría de la programación orientada a objetos se ha desarrollado.

Entre los lenguajes orientados a objetos se destacan los siguientes:

- ABAP
- ABL Lenguaje de programación de OpenEdge de Progress Software
- ActionScript
- ActionScript 3
- Ada
- C++
- C#
- Clarion
- Clipper (lenguaje de programación) (Versión 5.x con librería de objetos Class(y))

- D
- Object Pascal (Delphi)
- Gambas
- Harbour
- Eiffel
- Java
- JavaScript (la herencia se realiza por medio de la programación basada en prototipos)
- Lexico (en castellano)
- Objective-C
- Ocaml
- Oz
- R
- Perl (soporta herencia múltiple. La resolución se realiza en preorden, pero puede modificarse al algoritmo linearization C3 por medio del módulo Class::C3 en CPAN)
- PHP (a partir de su versión 5)
- PowerBuilder
- Python
- Ruby
- Smalltalk (Proyecto investigativo. Influenció a Java.)
- Magik (SmallWorld)
- Vala
- VB.NET
- Visual FoxPro (en su versión 6)
- Visual Basic 6.0
- Visual Objects
- XBase++
- Lenguaje DRP
- Lenguaje de programación Scala (lenguaje usado por Twitter)

Muchos de estos lenguajes de programación no son puramente orientados a objetos, sino que son híbridos que combinan la POO con otros paradigmas.

Al igual que C++ otros lenguajes, como OOCOBOL, OOLISP, OOPROLOG y Object REXX, han sido creados añadiendo extensiones orientadas a objetos a un lenguaje de programación clásico.

Un nuevo paso en la abstracción de paradigmas de programación es la Programación Orientada a Aspectos (POA). Aunque es todavía una metodología en estado de maduración, cada vez atrae a más investigadores e incluso proyectos comerciales en todo el mundo.

BIBLIOGRAFÍA

http://dis.um.es/~jfernand/0506/dai/poo.pdf

http://librosweb.es/libro/python/capitulo_5/programacion_orientada_a_objetos.html

http://www.ciberaula.com/articulo/tecnologia_orientada_objetos/

http://www.desarrolloweb.com/manuales/teoria-programacion-orientada-objetos.html