UNIVERSIDAD CATOLICA DE COLOMBIA
FACULTAD DE INGENIERIA DE SISTEMAS

CURSO: JAVA BASICO
PROFESOR: EMERSON CASTAÑEDA SANABRIA

TEMA: Programación Orientada a Objetos

OBJETIVOS:

- Familiarizarse con la Programación Orientada a Objetos (POO).
- Aprender los conceptos generales sobre la POO.
- Distinguir entre lenguajes de programación orientados a objetos y lenguajes de programación no orientados a objetos.

CONTENIDO:

1. Terminología
2. Características

DESARROLLO:

1. Terminología

Los términos fundamentales de la POO son, Clases, Objetos, Métodos, Atributos, Estados y Mensajes, a continuación se muestra la definición de cada uno con la finalidad de esclarecer la importancia de cada uno según el contexto de la POO en donde aplican.

Estos términos corresponden a los bloques de construcción que todo programador debe usar a fin de resolver problemas por medio de código orientado a objetos, con los cuales se debe estar profundamente familiarizado a fin de codificar con un mínimo de habilidad.

Clases

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos, o el anteproyecto de los objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando se programa un objeto y se definen sus características y funcionalidades en realidad lo que se esta haciendo es programar una clase.

Objetos

Consiste en cada uno de los elementos que a partir de la metodología orientada a objetos permite hacer representaciones de elementos, más cercanas al mundo real.
Los objetos son ejemplares de una clase cualquiera. Cuando se crea un ejemplar se debe especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar. 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.

```
miAuto = new Auto()
```

Con la palabra new se especifica que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis se pueden colocar parámetros con los que se inicializara el objeto de la clase Auto.

Métodos

Son las funcionalidades asociadas a los objetos. Cuando se programan las operaciones o funciones que puede llevar a cabo una clase, a estas se les llama los métodos de la clase. Es así como los métodos son las funciones que están asociadas a un objeto.

Atributos

Las propiedades o atributos son las características de los objetos. Cuando se define una propiedad normalmente se especifica su nombre y su tipo. Se puede decir que las propiedades son algo así como variables donde son almacenados los datos relacionados con los objetos.

Estados

Cuando se tiene un objeto sus propiedades o atributos toman valores. Por ejemplo, cuando se tiene un objeto auto 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.

```
miAuto.color = rojo
```

El objeto es miAuto, luego se coloca el operador punto y por último el nombre e la propiedad a la que se desea acceder. En este ejemplo se esta cambiando el valor del estado de la propiedad del objeto a rojo con una simple asignación.

Mensajes

Un mensaje en un objeto es la acción de efectuar una llamada a un método. Por ejemplo, cuando se le dice a un objeto auto que se ponga en marcha se le esta pasando el mensaje “ponte en marcha”. Para mandar mensajes a los objetos se utiliza el operador punto, seguido del método que desea invocar.

```
miAuto.ponerseEnMarcha()
```

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

2. Características

Existen cuatro características principales soportadas por la programación orientada a objetos:

Encapsulamiento

Corresponde con la forma de ocultar y proteger la información (también conocido como empaquetado) que se traduce en fácil desarrollo y mantenimiento de las aplicaciones construidas a partir de una metodología orientada a objetos.

El término indica la capacidad de los objetos de construir una cápsula a su alrededor que lo convierte en una caja negra, ocultando la información interna que manipulan. El encapsulamiento aparece ya en los lenguajes basados en procedimientos como una forma de proteger los datos. Con esto no hay que preocuparse, por ejemplo, si un nivel superior está usando una misma variable para una determinada iteración.

El encapsulamiento permite el uso de librerías de propósito general. Además, al verse como una caja negra, se puede, respetar sus entradas y salidas, modificar el código interno optimizándolo constantemente sin efectos laterales o secundarios.
La POO permite además encapsular las estructuras de datos que sirven como base a las funciones. Aportan, por tanto un nivel superior en cuanto a protección de información.

La encapsulamiento está en el núcleo de los dos grandes pilares de la construcción de sistemas: reusabilidad y mantenibilidad.

Polimorfismo

El polimorfismo es una de los nuevos conceptos aportado por la POO. Esta propiedad indica la posibilidad de definir varias operaciones con el mismo nombre. Dependiendo del objeto que se introduzca como entrada (receptor), se eligirá automáticamente cual de las operaciones (métodos) se va a realizar.

Polimórfico deriva de un vocablo griego y significa "muchas cosas".

Todos los lenguajes de programación proveen un operador suma. Sin embargo no existen en general los operadores sumaDeFracciones o sumaDeImaginarios. Los lenguajes de POO permiten definir un operador suma tal que reconozca a que tipo de objeto se le está aplicando. Previamente se deberá definir la clase de objeto Fracción e Imaginario y la operación suma. Internamente un fraccionario sabe como sumar un número (de cualquier tipo) a si mismo, y nos independizamos de su implementación para abstraernos al concepto de suma.

Este concepto es posible extenderlo a sumas entre, por ejemplo, bases de datos. Aquí la semántica de la suma es definida internamente por un manejador de bases de datos.

El error más común en el uso del polimorfismo es que en el afán de simplificar, se puede llegar a desvirtualizar el nombre del operador. Esto es, ¿Cumple el + de las bases de datos las propiedades conmutativa, asociativa, de elemento neutro, etc..? ¿Qué ocurre si sumo dos bases de datos con estructuras distintas?. Por esta razón se debe tener cuidado con el uso del polimorfismo sobre operadores de datos simples en estructuras complejas.

Dejando de lado la redefinición de operadores, una de las ventajas más importantes que ofrece el polimorfismo es que junto al encapsulamiento independizan completamente a las clases del resto del sistema. Con el encapsulamiento no importan, por ejemplo, los identificadores internos que utilicen otras clases y con el polimorfismo no interesan si otra clase usa el mismo nombre de método para identificar una operación ya que en ejecución se reconoce automáticamente a que clase se envía el mensaje (vinculación dinámica). Esta es una de las tantas cartas de triunfo que acreditan la POO.

Si se tiene un conjunto de objetos gráficos, como son, Circulo, Rectángulo, Rombo y se quiere que se dibujen, si no se cuenta con la capacidad de polimorfismo, la solución sería:

myCirculo dibujarCirculo
myRectangulo dibujar Rectangulo
mRombo dibujarRombo

Pero se cuenta con el polimorfismo, la solución consiste en enviarles a cada objeto el mensaje "dibujar".

La fuerza del polimorfismo radica en que permite que los algoritmos de alto nivel se escriban una vez y se reutilicen en forma repetida con diferentes abstracciones de bajo nivel.

Un concepto relacionado con el polimorfismo es la vinculación dinámica (binding dinámico) que significa que el enlace entre el objeto receptor del mensaje y el propio mensaje se realiza en forma dinámica en tiempo de ejecución.

Herencia

La herencia consiste en propagar los atributos y las operaciones a través de las distintas sub-clases definidas a partir de una clase en común.

Introduce una posibilidad de refinamiento sucesivo del concepto de clase.

La herencia permite crear estructuras jerárquicas de clases donde es posible la creación de subclases que incluyan nuevas propiedades y atributos que refinen los objetos. Estas subclases admiten la definición de nuevos atributos, así como de crear, modificar o inhabilitar propiedades para aumentar la especialización de la nueva clase.

Al poder reusar código definido por herencia, nos evitamos estar reinventando la rueda todo el tiempo. Gracias a esta característica, un software puede construirse en su mayor parte con componentes reutilizables lo que define al entorno como apto para el modelado rápido de prototipos.

Todos los modelos de un mismo fabricante de autos parten de un modelo básico al cual se le introducen distintas características (aire acondicionado, estéreo, levanta-vidrios electrónico, etc...) que van creando subclases del modelo original.

SubClase: Una clase que hereda el comportamiento y estructura interna de otra clase. Además adiciona su propio comportamiento para definir su tipo de objeto propio y único.

SuperClase: Una clase de la cual otras clases heredan su comportamiento y estructura interna.

La herencia es una de las propiedades más importantes de la POO, ya que permite a través de una clase básica ir añadiendo propiedades a medida que sean necesarias y haciendo más precisos los propios objetos.

También permite a los objetos compartir datos y comportamientos (heredarlos) a través de las subclases, sin incurrir en redundancia. Esto deriva en el ahorro de código y fundamentalmente en la claridad que aporta al identificar que en realidad las distintas operaciones sobre los objetos son una misma cosa.

Vinculación Dinámica

Es la base del polimorfismo y se refiere al hecho de que los objetos envían mensajes entre si, sin saber realmente su tipo especifico. Significa la búsqueda de métodos aplicables se decide en el momento de la ejecución.