Universidad Andrés Bello®
Conectar • Innovar • Liderar

FUNDAMENTOS DE PROGRAMACIÓN EN JAVA

El Paradigma de Orientación a Objeto

Importancia de la Orientación a Objetos en la Programación

La programación orientada a objetos (POO) es un paradigma de programación basado en objetos, los cuales manipulan los datos de entrada para la obtención de datos de salida específicos, y donde cada objeto ofrece una funcionalidad especial.

Estos objetos interactúan entre sí mediante procedimientos definidos (métodos) y vanevolucionando su información interna (propiedades) en función de su historia.

Si se hace una analogía, los objetos se parecen mucho a cómo funciona una empresa. Cada uno de los departamentos internos de la empresa como: ventas, administración, personal o producción, son objetos abstractos a los que el director solicita información (datos) o procesos (sacar las nóminas) sin que este conozca muy bien cuáles son los procedimientos internos a realizar.

El director tiene una imagen de la responsabilidad de cada sección (objeto) y de los procedimientos que cada objeto puede realizar, o de la información que le puede conseguir, y por eso el reporte de ventas por producto, sabe que corresponde a Ventas yno a nóminas, pero no se le ocurre pedir los datos concretos, sino solo el resumen procesado de acuerdo a ciertos criterios establecidos.

Además vale la pena destacar que cada departamento de la empresa dispone de sus propios datos internos privados, que utiliza para cumplir sus procedimientos, es decir, que al igual que la POO mezcla datos y procedimientos en el modelo de trabajo.

Son muchas las ventajas de una Programación Orientada a Objetos, pero las más relevantes son:

- **Modificabilidad:** en la POO es sencillo añadir, modificar o eliminar nuevos objetos o funciones que permiten actualizar programas fácilmente.
- **Gestión de los errores:** cuando se trabaja con un lenguaje POO se sabe exactamente dónde mirar cuando se produce un error, ventaja del trabajo modular de los lenguajes POO. Al poder dividir los problemas en partes más pequeñas se pueden probar de manera independiente, y así aislar los errores quepuedan producirse en el futuro.
- **Trabajo en grupo:** es más fácil trabajar en grupo ya que la POO permite minimizarla posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
- **Reducción de costes de programación:** especialmente en proyectos grandes la POO reduce los costos de programación, ya que los programadores pueden usar el trabajo de los otros, ahorrando horas de desarrollo. Crear librerías y compartirlas o reutilizar librerías de otros proyectos es algo habitual en la programación orientada a objetos.

Clases y Objetos

En el paradigma de la Programación Orientada a Objetos, los objetos son programas quetienen un estado y un comportamiento, conteniendo datos almacenados y tareas realizables durante su ejecución. Algunos ejemplos de objetos son: persona, silla, mesa, casa y auto.

Las clases son un pilar fundamental de la POO y representan un conjunto de variables y métodos para operar con datos. Una clase es una plantilla que define la forma de un objeto, especifica los datos y el código que operará sobre esos datos. Java usa una especificación de clase para construir objetos. Los objetos son instancias de una clase. Porlo tanto, una clase es esencialmente un conjunto de planes que especifican cómo construir un objeto.

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de determinado tipo. Por ejemplo, en un juego, una clase para representar a personas puedellamarse Persona y tener una serie de atributos como Nombre, Apellidos o Edad (que normalmente son propiedades), y una serie de comportamientos que pueden tener, como Hablar(), Caminar() o Comer(), y que se implementan como métodos de la clase (funciones).

Si se toma de ejemplo la clase Auto, en código Java quedaría de la siguiente manera:

```
class Auto {

}
```

Como se puede observar, el nombre de la clase va precedido por la palabra reservada class. Además, se incluyen llaves de inicio de bloque de la clase y otro de término, entendiendo que todo lo que se encuentre dentro de esas llaves será parte de la clase Auto.

Atributos de una clase y estado de un objeto

Una vez entendidos los conceptos de clase y objeto, se podrá continuar agregando propiedades de las que disponen los objetos que definen sus características individuales y le permiten diferenciarse de otros.

Los atributos son valores o características de los objetos. Permiten definir el estado del objeto u otras cualidades.

Retomando el ejemplo anterior de la clase Auto, se pueden reconocer algunos de sus posibles atributos: marca, color, capacidad de combustible, potencia del motor, velocidad, etc. Al llevar algunos de estos atributos a código Java quedaría de la siguiente manera:

```
class Auto {
    String marca;
    String color;
    float velocidad;
}
```

De esta forma, es posible crear la estructura de la clase con los atributos que corresponden y que permitirán diferenciar objetos.

Cabe recordar que una clase es siempre una simplificación de la realidad. Por esta razón,nunca se deben declarar atributos para todas y cada una de las características del objetoque se quiere representar; sólo se deben crear aquellos que se necesiten para resolver elproblema planteado. Por ejemplo, si se debe crear una clase Auto para guardar datos como marca, color y velocidad, no es necesario agregar atributos como: número de ruedas, modelo, capacidad de pasajeros o sistema de freno.

Métodos de una clase y Comportamiento de un objeto

Un método es una subrutina que puede pertenecer a una clase u objeto, y son una seriede sentencias para llevar a cabo una acción.

El objeto auto tiene los métodos: acelerar, frenar, indicar izquierda, indicar derecha, etc.Llevando a código Java algunos de los métodos señalados, se vería de la siguiente forma:

```
class Auto {
    String marca;
    String color;
    float velocidad;

    void acelerar() {
        velocidad++;
    }

    void frenar() {
        velocidad = 0;
    }
}
```

Para ambos ejemplos de métodos de la clase Auto, se modifica el atributo velocidad. Estoquiere decir que cualquier instancia Auto podrá utilizar estos métodos.

Métodos constructores

Existe un método que se ejecuta automáticamente cuando se instancia un objeto de unaclase. Este método se llama Constructor, el cual tiene por objetivo asignar los valores iniciales del nuevo objeto recién creado.

Un constructor de la clase Auto se vería de la siguiente forma en código Java:

```
class Auto {
    String marca;
    String color;
    float velocidad;

    void acelerar() {
        velocidad++;
    }
```

```
    void frenar() {
        velocidad = 0;
    }

    Auto() {
    }

}
```

Por ser métodos, los constructores también aceptan parámetros. Cuando en una clase no se especifica ningún tipo de constructor, el compilador añade uno por omisión sinparámetros, el cual no hace nada, por lo que los valores de los atributos de ese objeto tomarán los valores por defecto o con basura, según se haya escrito el código.

El mismo ejemplo de la clase Auto pero sumando un constructor con parámetros sería:

```
class Auto {
    String marca;
    String color;
    float velocidad;

    void acelerar() {
        velocidad++;
    }

    void frenar() {
        velocidad = 0;
    }

    Auto() {
    }

    Auto(String marca, String color, float velocidad) {
        this.marca = marca;
        this.color = color;
        this.velocidad = velocidad;
    }

}
```

Se debe tomar en consideración que los nombres de los parámetros de entrada no necesariamente deben tener el mismo nombre de los atributos de la clase. Hay algunas convenciones que sugieren utilizar el mismo nombre, como también otras que prefieren usar un guión bajo antes del nombre del atributo de la clase.

Cuando una función accede a un atributo lo hace en el contexto de un objeto. Por eso noes necesario que se especifique a qué objeto se está haciendo referencia. Sin embargo, hay casos en los que puede ser intex|x|resante disponer de dicho objeto, por ejemplo para pasarlo por parámetro a otra función o para distinguirlo de una variable o un parámetro homónimo.

Las funciones miembro pueden acceder al objeto actual mediante el objeto predefinido this.

En el caso del ejemplo de la clase Auto, se utiliza this para distinguir entre el atributo de la clase y el parámetro de la función. El “this.marca” hace referencia al atributo “marca” del objeto actual (this), mientras que “marca”, sin el this, hace referencia al parámetro dela función.

Modificadores de Acceso

Las clases tienen ciertos privilegios de acceso a los datos y a las funciones de otras clasesdentro de un mismo paquete. Los paquetes son una forma de organizar grupos de clases,contienen un conjunto de clases relacionadas. Los paquetes resuelven el problema del conflicto entre los nombres de las clases. Al crecer el número de clases crece la probabilidad de designar con el mismo nombre a dos clases diferentes. En Java, los paquetes se visualizan de la siguiente manera:

```
package nombrePaquete;

class Auto {

}
```

Para importar clases de un paquete se usa el comando import. En Java se puede importaruna clase individual al inicio de todo el código de la siguiente forma:

```
import java.util;

package nombrePaquete;

class Auto { }
```

Los modificadores de acceso determinan desde qué clases se puede acceder a un determinado elemento.

En Java existen 4 tipos de modificadores de acceso: public, private, protected yel tipo por default o por defecto. Si no se especifica ningún modificador de acceso seutiliza el nivel de acceso por defecto, que consiste en que el elemento puede ser accedidosólo desde las clases que pertenezcan al mismo paquete. Para este modificador de acceso, no se le debe agregar ninguna palabra reservada como en los otros casos.

public: Permite acceder al elemento desde cualquier clase, independientemente de que esta pertenezca o no al paquete en que se encuentra el elemento.

private: Es el modificador más restrictivo y especifica que los elementos que lo utilizan sólo pueden ser accedidos desde la clase en la que se encuentran.

protected: indica que los elementos sólo pueden ser accedidos desde su mismo paquete(como el acceso por defecto) y desde cualquier clase que extienda la clase en que se encuentra, independientemente de si esta se encuentra en el mismo paquete o no.

Los distintos modificadores de acceso quedan resumidos en la siguiente tabla:

Modificadores de acceso				
Modificador	**La misma clase**	**Otra clase del mismo paquete**	**Subclase de otro paquete**	**Otra clase de otro paquete**
default	X	X		
public	X	X	X	X
private	X			
protected	X	X	X	

Continuando con el ejemplo de la clase Auto visto en clases anteriores, los modificadoresde acceso de atributos y métodos quedarían de la siguiente forma:

```
class Auto {
    private String marca;
    private String color;
    private float velocidad;

    public void acelerar() {
        velocidad++;
    }
```

```
    public void frenar() {
        velocidad = 0;
    }

    public Auto() {
    }

    public Auto(String marca, String color, float
velocidad) {
        this.marca = marca;
        this.color = color;
        this.velocidad = velocidad;
    }
}
```

Los atributos poseen el modificador de acceso privateya que interesa salvaguardar los datos de los objetos. Por otro lado, los métodos son públicos para que pueda existir la comunicación entre distintas clases.

Accesadores y Mutadores

Si se observa el ejemplo anterior de la clase Auto, sus atributos son privados, por lo queotras clases no podrán acceder a los valores de éstos. Es por lo mismo que se necesitan métodos específicos para cumplir con esta misión.

El papel de los accesadores y mutadores es devolver y establecer los valores del estado de un objeto.

Quizás la conclusión más lógica sería cambiar los campos privados de la definición de la clase para que sean públicos y lograr los mismos resultados; sin embargo, es importante recordar que lo que se desea es ocultar los datos del objeto tanto como sea posible.

Accesador

Se utiliza un método accesador para devolver el valor de un campo privado. Sigue unesquema de nombres que prefija la palabra "get" al principio del nombre del método.

Por ejemplo, se agregarán métodos accesadores para los atributos marca, color y velocidadde la clase Auto:

```
class Auto {
    private String marca;
    private String color;
    private float velocidad;

    public void acelerar() {
        velocidad++;
    }

    public void frenar() {
        velocidad = 0;
    }

    public Auto() {
    }

    public Auto(String marca, String color, float
velocidad) {
        this.marca = marca;
        this.color = color;
        this.velocidad = velocidad;
    }

    public String getMarca() {
        return this.marca;
    }

    public String getColor() {
        return this.color;
    }

    public float getVelocidad() {
        return this.velocidad;
    }
}
```

Estos métodos siempre devuelven el mismo tipo de datos que su campo privadocorrespondiente y luego simplemente devuelven el valor de ese campo privado.

Mutador

Se utiliza un método de mutador para establecer el valor de un campo privado. Sigue un esquema de nombres con el prefijo "set" al comienzo del nombre del método. A continuacióm se agregarán los métodos mutadores para la marca, color y velocidad de laclase Auto:

```
class Auto {
        private String marca; private String color;
        private float velocidad;

        public void acelerar() {velocidad++;
        }

        public void frenar() {velocidad = 0;
        }

        public Auto() {
        }

        public Auto(String marca, String color, float
velocidad) {
                this.marca = marca; this.color = color;
                this.velocidad = velocidad;
        }

        public String getMarca() { return
                this.marca;
        }

        public String getColor() {return this.color;
        }

        public float getVelocidad() { return
                this.velocidad;
        }

        public void setMarca(String marca) {this.marca = marca;
        }

        public void setColor(String color) {this.color = color;
        }

        public void setVelocidad(float velocidad) {return this.velocidad;
        }
}
```

Estos métodos no tienen un tipo de retorno y aceptan un parámetro que es el mismo tipode datos que su campo privado correspondiente. El parámetro se utiliza para establecer el valor de ese campo privado.

Colaboración entre objetos

Es común que para resolver un problema de mediana complejidad utilizando la POO se utilice más de una clase, las cuales deben interactuar y comunicarse a través de los métodos.

Dentro de una aplicación Java, los objetos pueden estar conectados dentro de un programa con distintos tipos de relaciones. Estas relaciones pueden ser persistentes si establecen si la comunicación entre objetos se registra de algún modo y por tanto puedeser utilizada en cualquier momento. En el caso de relaciones no persistentes entonces elvínculo entre objetos desaparece tras ser empleado.

Esta colaboración se puede llevar a cabo mediante el establecimiento de relaciones entreclases o entre instancias (relación de asociación y relación todo-parte: agregación y composición).

Asociación

En una asociación, dos instancias A y B relacionadas entre sí existen de forma independiente. No hay una relación fuerte. La creación o desaparición de uno de ellos implica únicamente la creación o destrucción de la relación entre ellos y nunca la creacióno destrucción del otro. Por ejemplo, un cliente puede tener varios pedidos de compra o ninguno.

La relación de asociación expresa una relación, de una dirección o bidireccional, entre lasinstancias a partir de las clases conectadas. El sentido en que se recorre la asociación se denomina navegabilidad de la asociación. Cada extremo de la asociación se caracteriza por el rol o papel que juega en dicha relación el objeto situado en cada extremo.

La cardinalidad o multiplicidad es el número mínimo y máximo de instancias que pueden relacionarse con la otra instancia del extremo opuesto de la relación. El formato en el quese escribe es mínima..máxima, por ejemplo:

- ✓ 1: Sólo uno (por defecto).
- ✓ 0..1: Cero a uno.
- ✓ N..M: Desde N hasta M (enteros naturales).
- ✓ 0..*: Cero a muchos.
- ✓ 1..*: Uno a muchos.
- ✓ 1,5,9: Uno o cinco o nueve.

Todo-Parte

En una relación todo-parte una instancia forma parte de otra. En la vida real se dice que A está compuesto de B o que A tiene B. La diferencia entre asociación y relación todo- parte radica en la asimetría presente en toda relación todo-parte. En teoría se distingue entre dos tipos de relación todo-parte.

Agregación

La agregación es una asociación binaria que representa una relación todo-parte (pertenece a tiene un, es parte de). Por ejemplo, un centro comercial tiene clientes.

La composición es una agregación fuerte en la que una instancia "parte" está relacionada,como máximo, con una instancia "todo" en un momento dado, de forma que cuando un objeto "todo" es eliminado, también son eliminados sus objetos "parte". Por ejemplo: unrectángulo tiene cuatro vértices, un centro comercial está organizado mediante un conjunto de secciones de venta, etc.

A nivel práctico se suele llamar agregación cuando la relación se plasma mediantereferencias (lo que permite que un componente esté referenciado en más de un compuesto). Así, a nivel de implementación una agregación no se diferencia de una asociación binaria. Por ejemplo: un equipo y sus miembros. Por otro lado, se suele llamarcomposición cuando la relación se conforma en una inclusión por valor (lo que implica que un componente está como mucho en un compuesto, pero no impide que haya objetos componentes no relacionados con ningún compuesto). En este caso si se destruyeel compuesto se destruyen sus componentes. Por ejemplo: un ser humano y sus miembros.

Algunas relaciones pueden ser consideradas agregaciones o composiciones, en función del contexto en que se utilicen.

Composición

Es una forma fuerte de composición, donde la vida de la clase contenida debe coincidir con la vida de la clase contenedor. Los componentes constituyen una parte del objeto compuesto. De esta forma, los componentes no pueden ser compartidos por varios objetos compuestos.
La supresión del objeto compuesto conlleva la supresión de los componentes.

Un ejemplo de composición: Una empresa contiene muchos empleados. Un objetoEmpresa está a su vez compuesto por uno o varios objetos del tipo empleado. El tiempo de vida de los objetos Empleado depende del tiempo de vida de Empresa, ya que si no existe una Empresa no pueden existir sus empleados.

La composición es un tipo de relación dependiente en dónde un objeto más complejo es conformado por objetos más pequeños. En esta situación, la frase "Tiene un", debe tenersentido: El auto tiene llantas o el notebook tiene un teclado.

Anexo: Referencias

1.- Lenguaje Java y Entorno de Desarrollo
Referencia: http://www.jtech.ua.es/j2ee/2006-2007/doc/sesion01-apuntes.pdf