Programación Orientada a Objetos

TEMAS

Julio, 2014
Guadalajara, Jalisco. México.

A continuación se presentan los temas base para programar con objetos

POO

1.3 Objetos

Definición

El autor C. Thomas Wu en su libro "Introducción a la Programación Orientada a Objetos, Programación en Java" define a objeto como "cualquier cosa tanto tangible como intangible que se pueda imaginar" podemos definir nosotros a un objeto como una **porción delimitada de la realidad que tiene características las cuales se eligen sólo las que nos sirvan para la solución del problema**. Estas características se dividen en atributos y métodos y su finalidad respectivamente es definir qué es y para qué servirá dicho objeto. Estas características llamadas también propiedades se pueden encontrar a través de la abstracción. Tomando como abstracción el proceso mental de extraer lo esencial (características) de un objeto en el caso de la programación orientada a objetos para la solución del problema.

Los objetos entonces que podremos programar se pueden clasificar en tangibles (cosas) por ejemplo lápiz, intangibles (eventos) por ejemplo una tabla de multiplicar y conjunto de personas por ejemplo usuario.

Las características de los objetos se dividen en dos:

- Atributos: características físicas que me indican qué es el objeto y se obtienen de hacer una abstracción en primer nivel. Se puede decir que los atributos son datos que se utilizarán en los métodos

- Métodos: son las características de comportamiento, describen qué hace el objeto, se obtienen llevando acabo una abstracción a profundidad. Los métodos se puede decir que son los procesos que el objeto ejecutará.

Declaración

Los objetos los vamos a declarar dentro de la clase de tal manera que queden encapsulados (ver tema 1.4), y se ejecutarán ya sea en otra clase, en algún otro método de otro objeto o en alguna función estática como por ejemplo la función principal.

Sintaxis de la declaración de un objeto:

<TipoObjeto> <identificador>;

Sintaxis de la creación de un objeto:

<identificador> = new <constructor>;

Constructor

La metodología orientada a objetos se basa en los constructores y destructores, los cuales son funciones especiales que no tienen tipos de datos y se llaman igual que la clase.

Un constructor sirve para crear al objeto y su principal finalidad es inicializar los valores del objeto, el constructor también realiza ciertos proceso para crear al objeto que ya están programados en él mismo tales como reservar espacio en memoria, crear un formato lógico, activar una tabla de símbolos. Todo esto para que los atributos y métodos declarados en una clase se ejecuten como una sola entidad.

PROGRAMACIÓN DE OBJETOS

CÓMO PROGRAMAR OBJETOS

Para programar Objetos es necesario establecer los siguientes pasos:

- Declarar al objeto dentro de una clase, encapsulandolo en ella (encapsulameinto).

- Ocultar de la implementación los detalles de la programación del objeto (ocultamiento de información).

- Crear una instancia o variable de referencia para mandar llamar su comportamiento y de esta manera acceder al objeto como si fuera una sola entidad aunque este declarado de forma abstracta.

- Comunicar un objeto con otro (relación entre clases).

Destructor

La finalidad del destructor es borrar al objeto una vez que ya no se usa cuando se está ejecutando el programa por lo que libera memoria dinámica así como la memoria RAM que se pidió al construirse el objeto.

En el lenguaje de Java NO EXISTE el destructor, en su lugar se encuentra lo que se le conoce como "Recolector de Basura". El recolector de basura pasa periódicamente al estarse ejecutando el programa y revisa la memoria y los objetos que encuentra en desuso los marca para su posterior borrado. El programador no tiene control ni puede programar nada sobre el recolector de basura.

Si se quiere borrar físicamente al objeto es necesario programar la función finalize( ) del lenguaje de java, aunque esto no es muy recomendable porque se estaría modificando directamente la memoria y si no se tiene una buena experiencia con esto se pueden crear errores de desbordamiento de pila o mal uso de recursos físicos en la memoria.

Sintaxis Constructor

La sintaxis del constructor en el lenguaje de Java es la siguiente:

```
<tipoAcceso> <Identificador> ( [<parámetros>]) {
      <sentencia 1>;
           ...
      <sentencia n>;
}
```

Recordando que los parámetros son opcionales y el identificador del constructor debe ser el mismo que el de la clase.

Para conocer cuales son los tipos de acceso ver el tema 1.5

LLamada Constructor

Para mandar llamar (ejecutar) al objeto es necesario crear una variable de referencia a la cual también se le conoce como instancia esto porque como podemos recordar el objeto está declarado en la instancia entonces para mandarlo llamar es necesario instanciar la clase o que es lo mismo hacer una referencia a dicha clase.

La sintaxis para trabajar con la variable de referencia se divide en dos: la declaración y la creación, éstas se pueden escribir en una sola línea de código o en dos:

Declaración:
<TipoObjeto> <identificador>;

Creación:
<identificador> = new <Constructor>;

La sintaxis en una sola línea de código es:

<TipoObjeto> <identificador> = new <Constructor>;

Lo que pasa en la memoria al ejecutarse esta línea de código es que genera un espacio en memoria con el identificador que se declaró y lo que contiene es un apuntador que en un inicio apuntará a null y una vez que se manda llamar el constructor, ese apuntador apunta al objeto declarado en la clase.

1.3 Clase

Definición

Una clase es un conjunto de objetos con las mismas características o bien lo que se le conoce como una familia de objetos. La finalidad de la clase es ser una contenedora de objetos, es decir, es el elemento de la programación orientada a objetos que nos permite declarar una sola vez al objeto dentro de ella para después crear tantos objetos como sean necesarios y una vez creados mandar llamar su comportamiento.

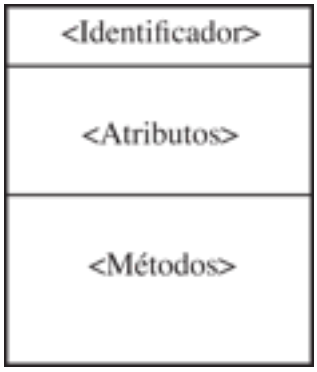
Simbología

La representación de una clase cuando se está desarrollando el diseño del sistema es un rectángulo con tres áreas.

La primera área es para establecer el identificador de la clase. Para seguir el estándar al identificador de la clase se le pone el mismo nombre que se va a declarar dentro de ella, es decir, si en la clase voy a declarar el objeto alumno entonces el identificador de la clase será "Alumno", esto no porque la clase y el objeto sean lo mismo sino porque como el objeto va a estar declarado de forma abstracta (enlistado de atributos y métodos).

La segunda área es para establecer los atributos y aunque se pueden poner solamente sus identificadores se recomienda de una vez indicar el tipo de dato de cada atributo así como su tipo de acceso de una vez.

La tercera área será para los métodos y de igual manera se puede establecer de una vez su tipo de dato y su tipo de acceso, cabe mencionar que aquí los atributos sólo se mencionarán.



Es importante mencionar que cuando se está diseñando la clase no hay una sintaxis que seguir, es decir dentro del área de atributos y métodos se pueden establecer detalles como tipos de datos, tipos de acceso, inicializaciones, parámetros para el caso de los métodos o simplemente establecer el identificador.

Por esta razón, los atributos siempre se escribirán en la segunda área del esquema y los métodos en la tercera área del esquema si se llega a cambiar este orden estará mal representado.

Declaración

La sintaxis de una clase se define de la siguiente manera:

```
[<tipoAcceso>] class <Identificador> {
        <tipoAcceso> <atributo1>;
            ...
        <tipoAcceso> <atributoN>;
        <tipoAcceso > <método1>( [<parámetros>] ) {
            <sentencias>;
        }
            ....
        <tipoAcceso > <métodoN>( [<parámetros>] ) {
            <sentencias>;
        }
}
```

Para conocer cuales son los tipos de acceso ver el tema 1.5

1.4 Encapsulamiento

Definición

Fuera de programación cuando se dice que algo se encapsula es porque estamos diciendo que será guardado en un lugar que esté cerrado herméticamente, es decir totalmente sellado en donde no entra ni sale nada. Esta misma definición usaremos para la programación orientada a objetos pero lo que estaremos encerrando serán los objetos y el lugar donde los encerramos herméticamente es en la clase. Por lo que al declarar atributos y métodos (objeto abstracto) dentro de la clase lo estamos encapsulando en ella por lo que debemos asegurarnos que podrá salir de ella. Para esto es necesario crear una instancia o variable de referencia como se explicó en el tema 1.1 (llamada del objeto) para ejecutar al objeto.

El encapsulamiento por lo tanto nos permite manejar a los atributos y métodos declarados en la clase como una sola entidad y no como datos y procesos por separado. Sin embargo, si se mandan llamar cualquier atributo y/o método a través de la instancia provocaría el rompimiento del encapsulamiento ya que estaríamos ejecutando sus características por separado. Para no romper con el encapsulameinto se utiliza el ocultamiento de información

Ocultamiento de Información

Significa esconder de la implementación (lugar donde se manda llamar el comportamiento del objeto, por lo general es la función principal) los detalles de la programación de dicho objeto. Esto significa que los atributos sólo se llamaran en sus propios métodos y los métodos que pertenezcan a dichos detalles, mandando llamar con la instancia los métodos que tengan el resultado deseado. Los tipos de acceso o control de acceso ayudan mantener este control

De esta manera logramos que un objeto declarado en una clase se comunique con otro objeto declarado en otra clase.

1.5 Tipos de Acceso

También se le conoce como control de acceso y nos permite llevar a cabo de una manera más precisa el ocultamiento de información y de esta manera asegurarnos de no romper con el encapsulamiento cuando los objetos se estén comunicando entre sí.

Cabe mencionar que los tipos de acceso no son sinonimos de ocultamiento de información, sólo ayudan a controlarlo.

Los tipos de acceso o control de acceso se pueden utilizar tanto para la clase como para los atributos y métodos pero se estará hablando de controles totalmente diferentes. El control de acceso para las clases nos permite definir el tipo de clase con la que vamos a trabajar mientras que el control de acceso para los atributos y métodos nos permite indicar cual será el alcance de los mismos.

Tipos de acceso para las clases:

público: su palabra reservada es public y significa que esa clase es accesible desde cualquier paquete.

abstracto: la palabra reservada es abstract y significa que la clase va a tener métodos abstractos que podrán ser implementados en otras clases. La clase abstracta no se instancia.

final: se establece la palabra reservada final y significa que esa clase no puede tener herencia.

Cuando no se pone tipos de acceso en las clases, no existe un tipo por default, esto quiere decir que esa clase puede ser accedida sólo por el paquete en donde se encuentre declarada.

Tipos de acceso para los atributos y métodos:

público: La palabra reservada es public y en el diseño se puede indicar con el símbolo de + (mas). Significa que se pueden mandar llamar (ser vistos) desde cualquier clase en cualquier archivo y en cualquier paquete, es decir, son accesibles desde cualquier lugar.

privado: La palabra reservada es private y en el diseño se puede indicar con el símbolo de - (menos) significa que se pueden mandar llamar (ser vistos) sólo desde la misma clase, es decir, únicamente los métodos declarados en esa clase pueden acceder a los atributos y métodos privados.

protegido: La palabra reservada es protected y en el diseño se puede indicar con el símbolo de # (gato). Significa que sólo se pueden mandar llamar (ser vistos) por las subclases. Las subclases son clases que dependen de otra clase (herencia).

amigable: Es el tipo de acceso por default, es decir si no se establece ningún tipo de acceso, se convierten en friendly, no se tiene una palabra reservada en el lenguaje y tampoco un símbolo específico. Significa que pueden mandarse llamar (ser vistos) desde cualquier lugar siempre y cuando sean parte del mismo paquete.

Un paquete en java es un conjunto de clases.