Unidad VI

Modularidad

6.1 Declaración de métodos.

En programación modular, y más específicamente en programación orientada a objetos, se denomina Modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

Estos módulos que se puedan compilar por separado, pero que tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la Modularidad de diversas formas.

Según Bertrand Meyer ”El acto de particionar un programa en componentes individuales para reducir su complejidad en algún grado. . . . A pesar de particionar un programa es útil por esta razón, una justificación más poderosa para particionar un programa es que crea una serie de límites bien definidos y documentados en el programa. Estos límites, o interfaces, son muy valiosos en la comprensión del programa”

Por su parte Bárbara Liskov establece que “modularización consiste en dividir un programa en módulos que pueden ser compilados de forma separada, pero que tienen conexiones con otros módulos”.

La declaración mínima sin modificadores de un método es:

Donde:

- *TipoDevuelto* es el tipo de dato devuelto por el método (función). Si el método no devuelve ningún valor, en su lugar se indica la palabra reservada void.
- *NombreMetodo* es un identificado válido en Java.
- *Lista_Parametros* si tiene parámetros, es una sucesión de pares **tipo – valor** separados por comas. Los parámetros pueden ser también objetos.

Los tipos simples de datos se pasan siempre por valor y los objetos y vectores por referencia

Cuando se declara una subclase, esa subclase hereda, en principio, todos los atributos y métodos de la superclase (clase padre). estos métodos pueden ser redefinidos en la clase hija simplemente declarando métodos con los mismos identificadores, parámetros y tipo devuelto que los de la superclase. Si desde uno de estos métodos redefinidos se desea realizar una llamada al método de la superclase, se utiliza el identificador de la superclase y se le pasan los parámetros.

6.2 Métodos de clase.

En la programación orientada a objetos, un **método** es una subrutina asociada exclusivamente a una clase (llamados **métodos de clase** o **métodos estáticos**) o a un objeto (llamados **métodos de instancia**). Análogamente a los procedimientos en los lenguajes imperativos, un método consiste generalmente de una serie de sentencias para llevar a cabo una acción, un juego de parámetros de entrada que regularán dicha acción y o, posiblemente, un valor de salida (o valor de retorno) de algún tipo.

Algunos lenguajes de programación asumen que un método debe de mantener el invariante del objeto al que está asociado asumiendo también que éste es válido cuando el método es invocado. En lenguajes compilados dinámicamente, los métodos pueden ser objetos de primera clase, y en este caso se puede compilar un método sin asociarse a ninguna clase en particular, y luego asociar el vínculo o contrato entre el objeto y el método en tiempo de ejecución. En cambio en lenguajes no compilados dinámicamente o tipados estáticamente, se acude a precondiciones para regular los parámetros del método y postcondiciones para regular su salida (en caso de tenerla). Si alguna de las precondiciones o postcondiciones es falsa el método genera una excepción. Si el estado del objeto no satisface la invariante de su clase al comenzar o finalizar un método, se considera que el programa tiene un error de programación.

La diferencia entre un procedimiento (generalmente llamado *función* si devuelve un valor) y un método es que éste último, al estar asociado con un objeto o clase en particular, puede acceder y modificar los datos privados del objeto

correspondiente de forma tal que sea consistente con el comportamiento deseado para el mismo. Así, es recomendable entender a un método no como una secuencia de instrucciones sino como la forma en que el objeto es útil (el método para hacer su trabajo). Por lo tanto, podemos considerar al método como el pedido a un objeto para que realice una tarea determinada o como la vía para enviar un mensaje al objeto y que éste reaccione acorde a dicho mensaje.

6.3 Métodos de instancia.

Cuando una declaración de método incluye un modificador static, se dice que el método es un método estático. Si no existe un modificador static, se dice que el método es un método de instancia.
Un método estático no opera en una instancia específica, y se produce un error en tiempo de compilación al hacer referencia a this en un método estático.
Un método de instancia opera en una determinada instancia de una clase y es posible tener acceso a dicha instancia con this
Cuando se hace referencia a un método en un *acceso-a-miembro* de la forma E.M, si M es un método estático, E debe denotar un tipo que contenga M, y si M es un método de instancia, E debe denotar una instancia de un tipo que contenga M.