Instituto Politécnico Nacional

Escuela Superior de Cómputo

Programación Orientada

a Objetos

Niveles de Acceso

en Java

Profesor: Roberto Tecla Parra

Alumno: Calva Hernández José Manuel

Grupo: 2CM3

# Niveles de Acceso

Los niveles de acceso son las restricciones que tiene un dato para ser accedido. Estos niveles de acceso se dan tanto en las propias clases, como en los objetos instanciados de ellas, como sus métodos y atributos.

Se puede utilizar un especificador de acceso, llamado *Access* para fijar la visibilidad de los miembros de datos de una clase a lo largo del resto del programa.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ubicación | *Private* | *No modifier* | *Protected* | *Public* |
| Misma clase | Sí | Sí | Sí | Sí |
| Subclase del mismo paquete |  | Sí | Sí | Sí |
| No subclase del mismo paquete | Sí | Sí | Sí |  |
| Subclase de otro paquete |  |  | Sí | Sí |
| No subclase de otro paquete |  |  | Sí |  |

Los valores posibles de *Access* son *public, private y protected.* Cuando se declara un miembro de una clase como *public*, es accesible desde cualquier lugar del programa. Si se declara como *private*, sólo es accesible desde la clase de la que es miembro. Si se declara como *protected*, está disponible para la clase actual, otras clases del mismo paquete (se pueden agrupar librerías de clases en paquetes Java), y clases que son derivadas de esa clase. Si no se usa un especificador de acceso, el acceso por defecto es que el miembro de la clase es visible a la clase de la que es miembro, a las clases derivadas de la misma que están en su mismo paquete y a otras clases del mismo paquete.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modifier | Class | Package | Subclass | World |
| *Public* | Yes | Yes | Yes | Yes |
| *Protected* | Yes | Yes | Yes | No |
| *No modifier* | Yes | Yes | No | No |
| *Private* | Yes | No | No | No |

# Acceso a las variables

El valor de una variable de clase es compartido por todos los objetos de esa clase, lo que significa que será el mismo para todos los objetos. Una variable se declara como estática con la palabra clave *static* (que realmente especifica la forma en que el valor es almacenado, como dato estático).

El modificador *static* no sirve para crear constantes, sino para crear miembros que pertenecen a la clase, y no a una instancia de la clase. Esto implica, entre otras cosas, que no es necesario crear un objeto de la clase para poder acceder a estos atributos y métodos. Este es el motivo por el cual es obligatorio que *main* se declare como *static*; de esta forma no tenemos que ofrecer un constructor vacío para la clase que contiene el método, o indicar de alguna forma a la máquina virtual cómo instanciar la clase.

*[Access] static tipo* nombre\_de\_variable;

# Una característica no muy conocida que se introdujo en Java 1.5 son los *static imports*, una sentencia similar al *import* habitual, con la salvedad de que esta importa miembros estáticos de las clases, en lugar de clases de los paquetes, permitiendo utilizar estos miembros sin indicar el espacio de nombres en el que se encuentran.

Tanto *static* como otros modificadores afectan a las variables u objetos, sin embargo, no de la misma forma en que son afectados por los modificadores de acceso. El más parecido en este sentido, es el *static*, por lo cual fue descrito anteriormente.

# Acceso a los métodos

Uno de los aspectos más importantes de los métodos es que se puede hacer que sean puramente internos a un objeto, con el concepto de encapsulación de la programación orientada a objetos. Para declarar y definir un método, se puede usar un especificador de acceso y especificar el tipo de retorno del método si se quiere que devuelva un valor.

*[Access] class* nombre\_de\_clase [extends…] [implements…]

{

*[Access] [static] tipo* nombre\_de\_método\_1 (lista de parámetros)

{

…

}

…

*[Access] [static] tipo* nombre\_de\_método\_n (lista de parámetros)

{

…

}

}

Se puede restringir el acceso a los datos de sus objetos usando métodos de acceso a datos que deben ser invocados para obtener los datos. Usualmente se les conoce como *getters (getData) y setters (*setData*)*. El método getData devuelve el valor de la variable privada, mientras que el método setData restringe el acceso a los datos internos.

Es buena idea utilizar los métodos de acceso a datos para garantizar el acceso a los datos internos de los objetos. Usando estos métodos se puede controlar la interfaz con esos datos y por lo tanto bloquear operaciones que se consideren ilegales.