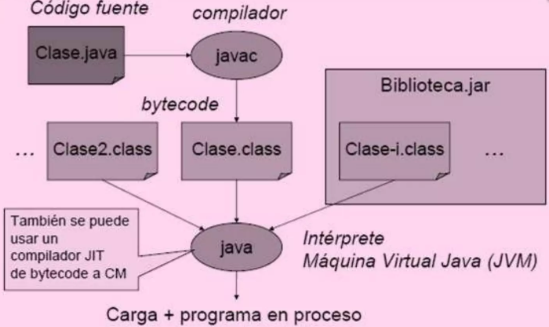
**Resumen Conceptos Básicos JAVA**



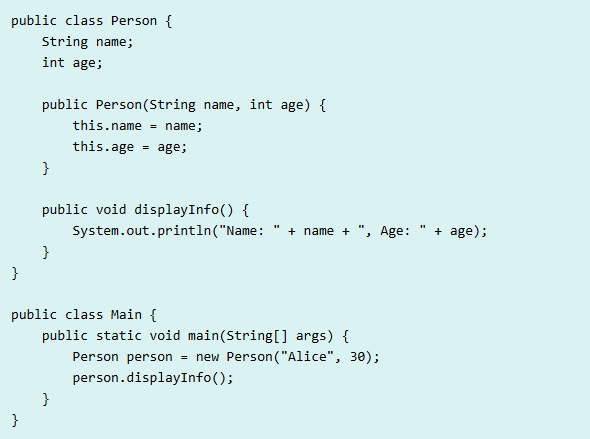
Java es un lenguaje de programación orientado a objetos y basado en clases. Fue diseñado para tener la menor cantidad de dependencias de implementación posibles, lo que lo hace ideal para aplicaciones que necesitan ser ejecutadas en múltiples plataformas. **Solo se necesita de una JVM para poder ejecutarse.**



**Tipo de datos primitivos**

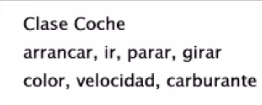
* byte: 8 bits, rango de -128 a 127
* short: 16 bits, rango de -32,768 a 32,767
* int: 32 bits, rango de -2^31 a 2^31-1
* long: 64 bits, rango de -2^63 a 2^63-1
* float: 32 bits, punto flotante
* double: 64 bits, punto flotante
* char: 16 bits, carácter Unicode
* boolean: true o false

**Clase:** Define la forma, y el comportamiento de un objeto, de un miembro real o una entidad abstracta y atributos de un grupo de objetos de características similares. En simples palabras una clase es una "plantilla" que describe un conjunto de objetos con atributos y comportamiento similares.

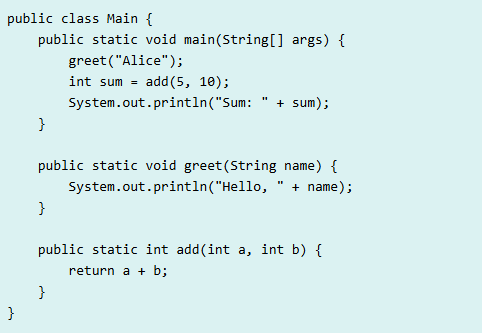


**Estructura de una clase**

* Nombre clase
* Atributos
* Métodos



**Métodos:** Los métodos son bloques de código que realizan una tarea específica y pueden ser llamados desde otras partes del programa.

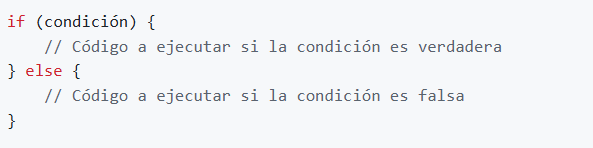


**Estructuras de Control**

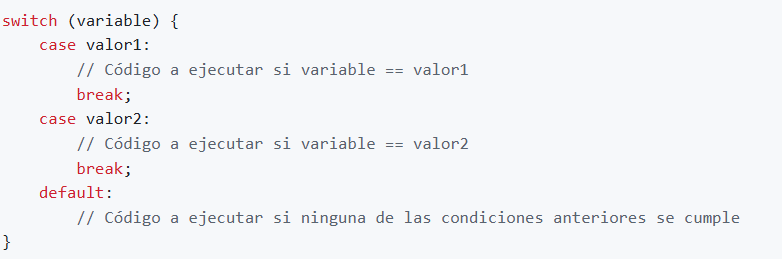
Las estructuras de control en Java se utilizan para gestionar el flujo de ejecución del programa. Las más comunes son.

Condicionales

If



Switch

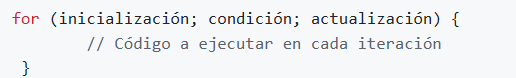


Operador ternario

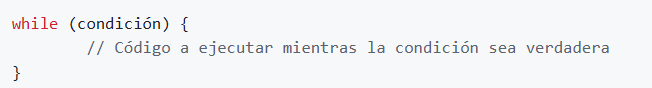


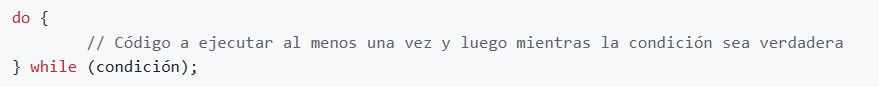
Bucles

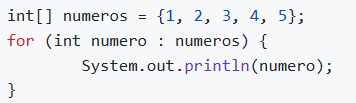
for



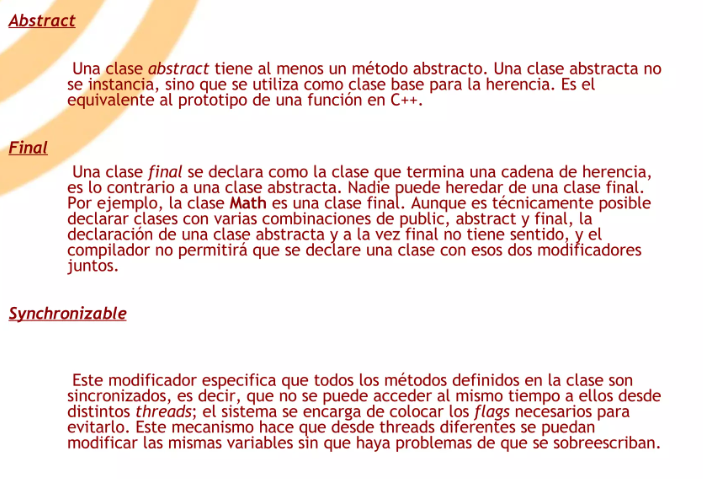
while

  
 do-while

  
Foreach



**Modificadores de clase**



**Creación de objetos**

Antes de poder usar un objeto hemos de crearlo. EI operador **new** nos permite crear objetos en Java.

Tipo identificador = new Tipo () ;

Tanto en Java como en C++, la creación de un objeto se realiza en tres pasos (aunque se pueden combinar):

• Declaración (proporcionar un nombre al objeto)

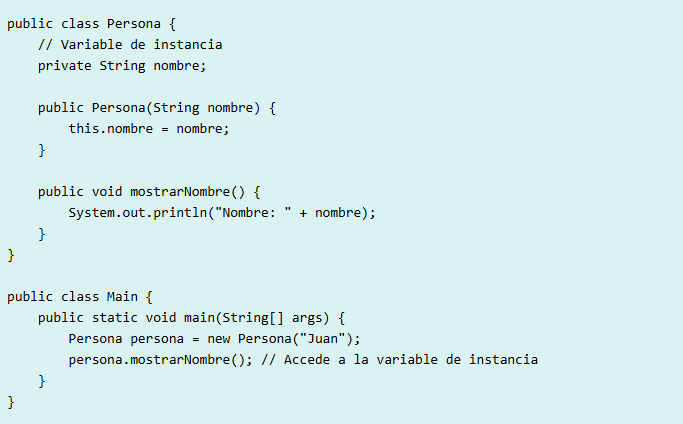
• Instanciación (asignar memoria al objeto)

• Inicialización (opcionalmente se pueden proporcionar valores iniciales a las variables de instancia del objeto)

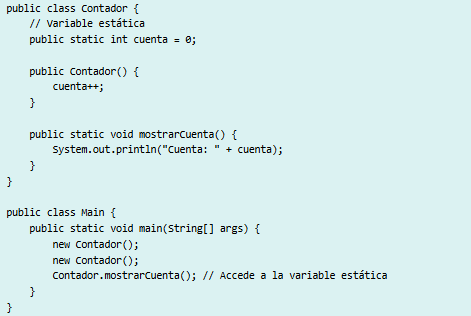
**Variables**

En Java, no existen variables globales como tal. Sin embargo, las variables de instancia y las variables estáticas pueden actuar de manera similar. Las variables de instancia pertenecen a un objeto y se pueden acceder desde cualquier método de la clase. Las variables estáticas pertenecen a la clase y se comparten entre todos los objetos de esa clase.

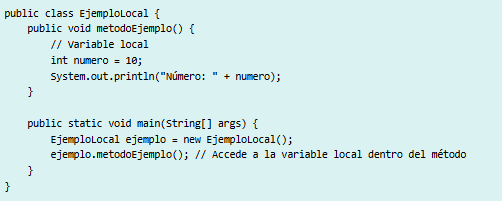
**Variables de Instancia.**



**Variables Estáticas**

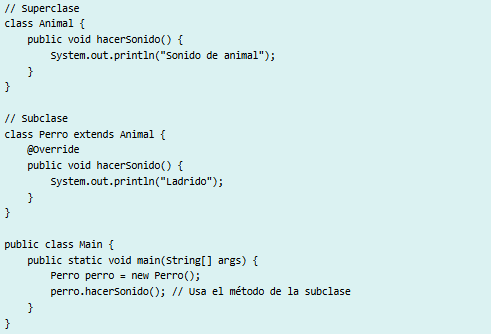


**Variables Locales** Son aquellas definidas dentro de un método, constructor o bloque y solo son accesibles dentro de ese contexto. Se crean cuando se invoca el método y se destruyen cuando el método termina.



**Herencia**

Es un mecanismo que permite crear una nueva clase a partir de una clase existente. La nueva clase (subclase) hereda los campos y métodos de la clase existente (superclase). Java soporta herencia simple, lo que significa que una clase solo puede extender otra clase



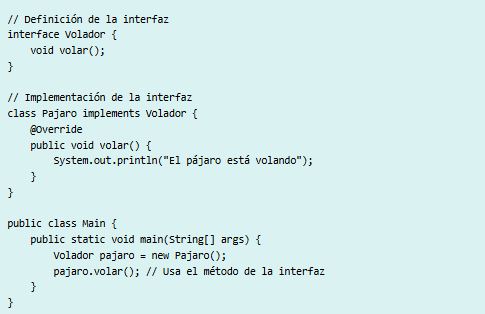
**Limitaciones en la herencia**

Todos los campos y métodos de una clase son siempre accesibles para el código de la misma clase. Para controlar el acceso desde otras clases, y para controlar la herencia por las subclases, los miembros (atributos y métodos) de las clases tienen tres modificadores posibles de control de acceso:

* **public:** Los miembros declarados public son accesibles en cualquier lugar en que sea accesible la clase, y son heredados por las subclases.
* **private:** Los miembros declarados private son accesibles sólo en la propia clase.
* **protected:** Los miembros declarados protected son accesibles sólo para sus subclases

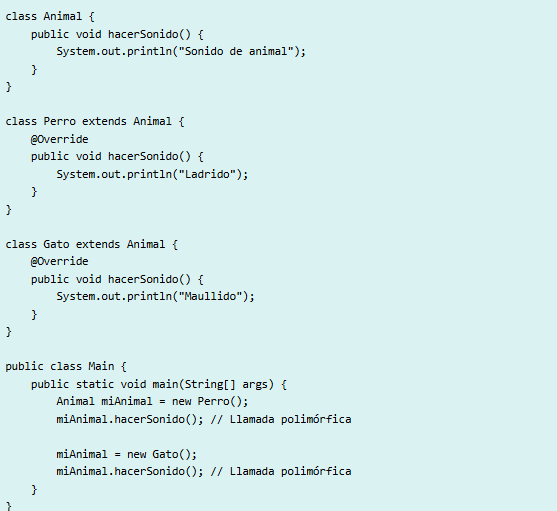
**Interfaces**

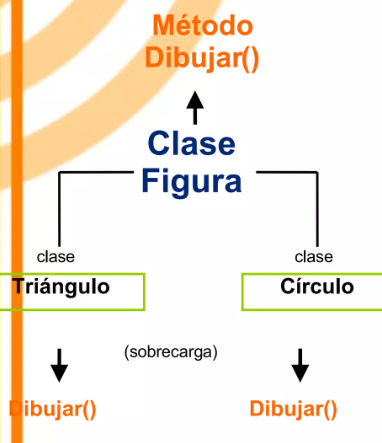
Una interfaz en Java es un tipo de referencia que es similar a una clase, pero solo puede contener métodos abstractos (sin implementación) y constantes. Las clases que implementan una interfaz deben proporcionar la implementación de los métodos definidos en la interfaz. Las interfaces permiten la implementación de múltiples herencias.



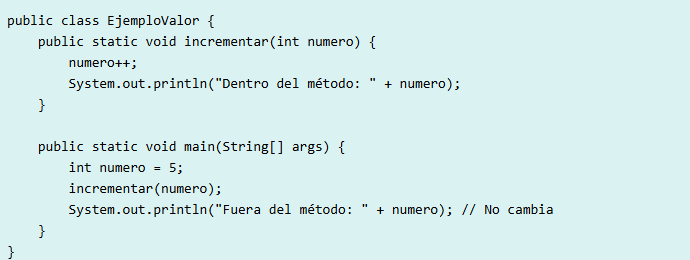
**Polimorfismo**

Es la capacidad de un objeto de tomar muchas formas. En Java, el polimorfismo se logra principalmente a través de la herencia y las interfaces. Permite que una sola acción se comporte de manera diferente en diferentes contextos, por ejemplo, el mismo método puede tener diferentes implementaciones en diferentes clases.

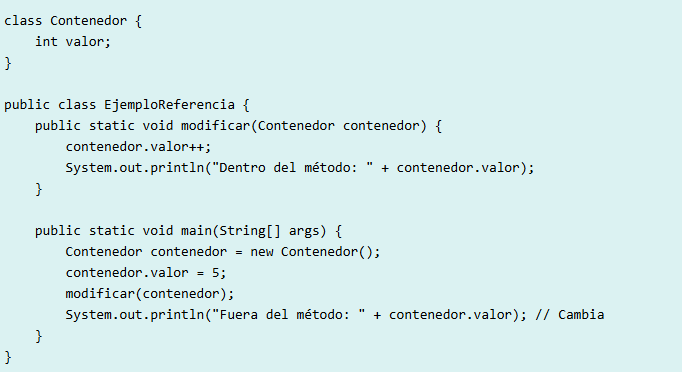




**Parámetros por Valor**: En Java, los tipos primitivos (int, char, etc.) se pasan por valor, lo que significa que se pasa una copia del valor al método.



**Parámetros por Referencia:** Los objetos se pasan por referencia, lo que significa que se pasa una referencia al objeto, no el objeto en sí. Esto permite que los métodos modifiquen el objeto original

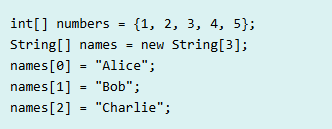


Colecciones en Java

Java proporciona una serie de interfaces y clases para manejar colecciones de objetos. Las principales interfaces de colecciones son List, Set, y Map. Además, los arrays son una estructura de datos fundamental en Java.

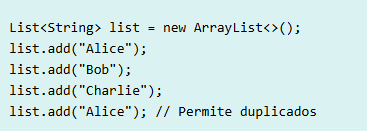
Array

Un array es una estructura de datos que contiene una secuencia de elementos del mismo tipo. Los arrays tienen un tamaño fijo que se define cuando se crea el array.



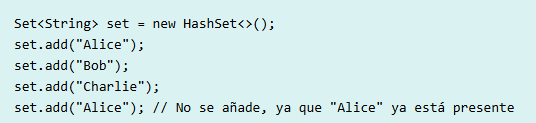
List

List es una colección ordenada que permite elementos duplicados. Las implementaciones más comunes son ArrayList y LinkedList.



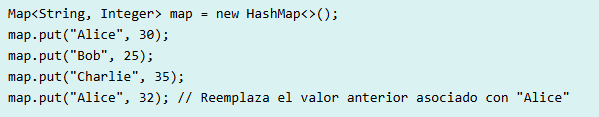
Set

Set es una colección que no permite elementos duplicados. Las implementaciones más comunes son HashSet, LinkedHashSet, y TreeSet.



Map

Map es una colección que mapea claves a valores. No permite claves duplicadas, pero sí valores duplicados. Las implementaciones más comunes son HashMap, LinkedHashMap, y TreeMap.



Genéricos en Java

Los genéricos permiten definir clases, interfaces y métodos con tipos parametrizados. Esto proporciona una mayor seguridad de tipos y elimina la necesidad de castings explícitos.

