Programación & Laboratorio III

Profesor Gonzalo Abel Benoffi
FRUTN - MDP - AÑO 2020

Evolución de la Programación

- Programación Lineal
- Programación Estructurada
- Programación Orientada a Objetos <- usted está aquí

Evolución de la Programación

- Programación Lineal

"Conjunto de técnicas matemáticas que intentan resolver la maximización o minimización de una función objetivo"

Evolución de la Programación

- Programación Estructurada

->Dividir el programa en segmentos

->Secuencia, selección e iteracción

Evolución de la Programación

- Los programas son más fáciles de entender, pueden ser leídos de forma secuencial y no hay necesidad de tener que rastrear saltos de líneas (GOTO) dentro de los bloques de código para intentar entender la lógica interna.
- La estructura de los programas es clara, ya que las instrucciones están más ligadas o relacionadas entre sí.
- Se optimiza el esfuerzo en las fases de pruebas y depuración. El seguimiento de los fallos o errores del programa (debugging), y con él su detección y corrección, se facilita enormemente.
- Los programas son más sencillos y más rápidos de confeccionar.
- Se incrementa el rendimiento de los programadores.

Programación Orientada a Objetos

Clasificación de los lenguajes orientados a objetos

- Lenguajes **basados en objetos** que soportan objetos Es decir, disponen de componentes caracterizados por un conjunto de operaciones (comportamiento) y un estado. Encapsulamiento + identidad de los objetos

Clasificación de los lenguajes orientados a objetos

- Lenguajes **basados en clases** que implican objetos y clases Es decir, disponen de componentes tipo clase con operaciones y estado común. Una clase de un objeto se construye con un «interfaz» que especifica las operaciones posibles y un «cuerpo» que implementa dichas operaciones.

  Basado en objetos + abstracción de conjunto

Clasificación de los lenguajes orientados a objetos

- Lenguajes **orientados a objetos** que además de objetos y clases ofrecen mecanismos de herencia entre clases. Esto es, la posibilidad de derivar operaciones y atributos de una clase (superclase) a sus subclases.

(NOT) Programación Orientada a Objetos

- Un sistema de ventanas, iconos, botones, imágenes
- Un lenguaje de programación como Java, C, C#, Android
- Un IDE como Eclipse, NetBeans, CodeBlocks

Programación Orientada a Objetos

“Primero resuelve el problema.

Entonces escribe el código.”

- John Johnson - Programador estadounidense.

Programación Orientada a Objetos

- Una forma de ver las cosas.
- Entender un problema identificando las principales entidades que se encuentran en él.
- Una forma de desarrollar un sistema pensando en las entidades principales del problema que dicho sistema pretende resolver.
- Su propósito es facilitar una solución (informática) identificando los conceptos o entidades relevantes presentes en el problema.

Programación Orientada a Objetos - Elementos concretos

Objetos

Clases

Herencia simple y múltiple

Comprobación estricta de tipos frente a comprobación débil

Encapsulamiento

Genericidad

Paso de mensajes

Polimorfismo

Excepciones

Concurrencia

Persistencia

Metadatos

Bibliotecas

IDEs

Administración de memoria

Programación Orientada a Objetos - Requisitos Deseables

- Conceptos claros
- Orientación al objeto pura
- Seguridad
- Alto nivel
- Sintaxis fácil de leer
- Eliminación de la redundancia
- Pequeño

Identificar los conceptos relevantes presentes en el problema.

- Reconocer las características.
- Reconocer las acciones que realizan.
- Reconocer cómo se comunican con otros objetos iguales o diferentes.
- Reconocer sus responsabilidades y alcance.

Fundamentos de la POO

- Abstracción
- Encapsulamiento
- Polimorfismo
- Herencia
- Modularidad

Abstracción

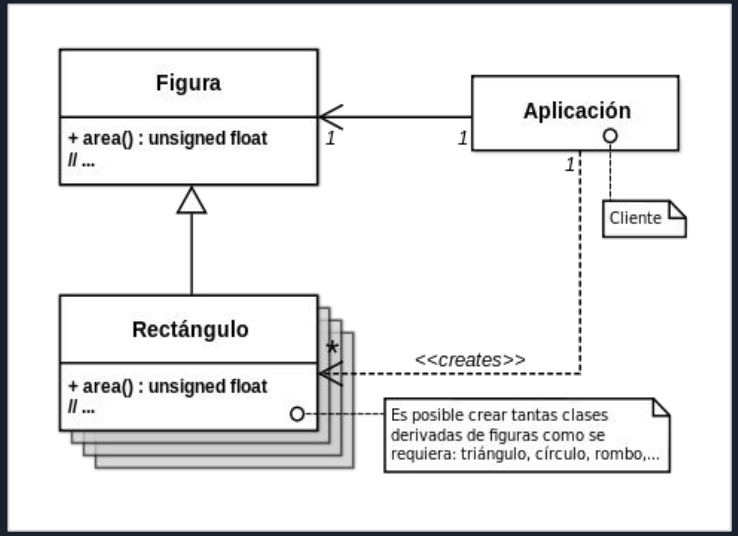
- Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar "cómo" se implementan estas características
- El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real.
- La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

Encapsulamiento

- Las variables de un objeto se localizan en el núcleo del objeto. Los métodos rodean y esconden al núcleo del objeto de otros objetos en el programa.
- Típicamente, el encapsulamiento es utilizado para esconder detalles de la puesta en práctica no importantes de otros objetos. Entonces, los detalles de la puesta en práctica pueden cambiar en cualquier tiempo sin afectar otras partes del programa.

Polimorfismo

- Un mismo método -o acción- puede tener el mismo nombre pero un comportamiento diferente para el mismo o diferentes objetos.
- Se diferencian con la cantidad y tipos de parámetros de entrada pero el retorno (y el nombre) es el mismo. (JAVA)
- El compilador elige el método a invocar en tiempo de ejecución.



Herencia

Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación.

Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes.

Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles.

Modularidad

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

#EstoLoTengoQueSaberParaElParcial

¿Por dónde empezamos?

Pues, por el principio.

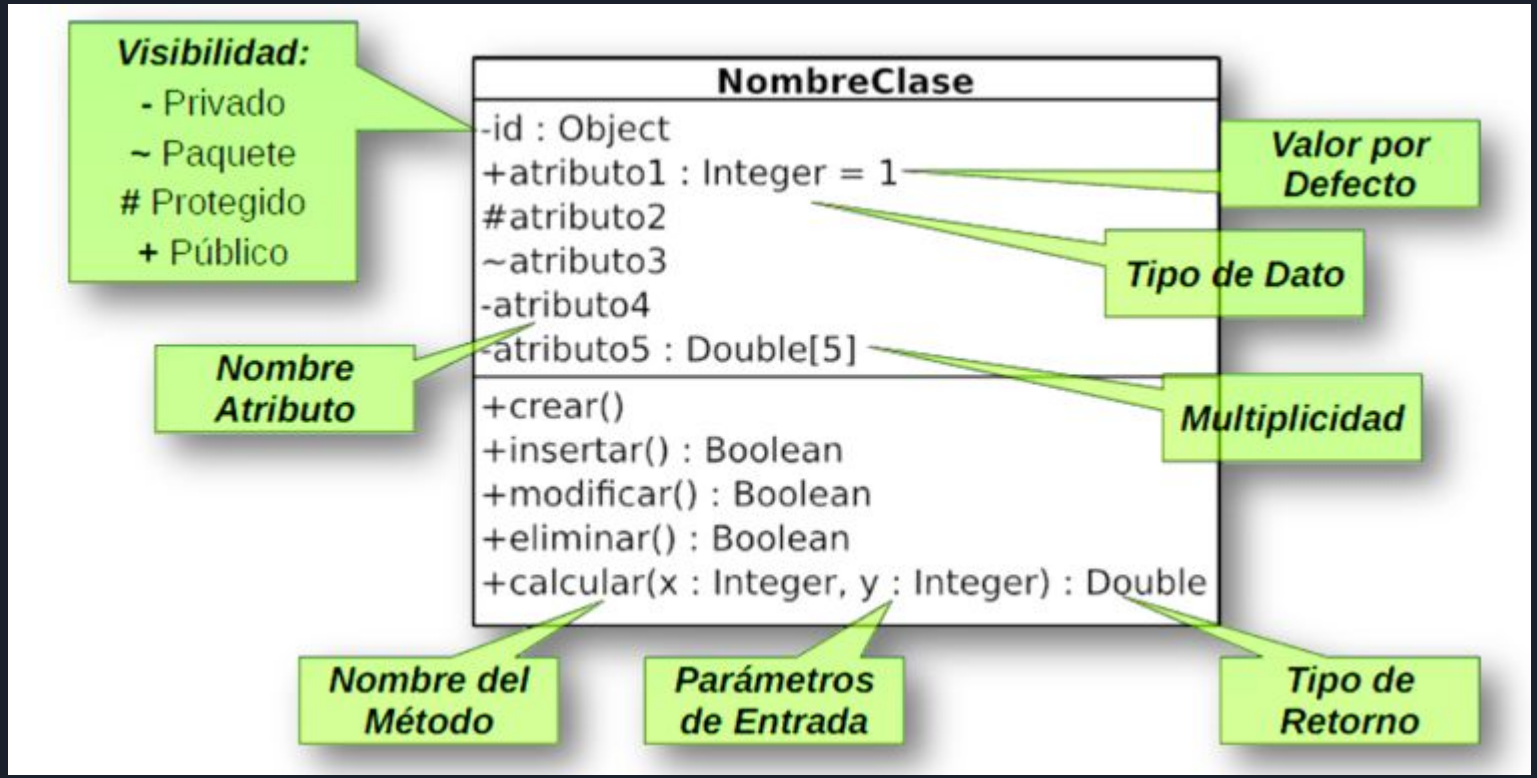
Comienzo del diseño de clases

- Encontrar clases, añadirles métodos y luego atributos o datos de la clase.
- Para identificar clases hay que buscar nombres en el análisis del problema. Los métodos corresponden a verbos.

¿Qué es una clase?

- Una plantilla, una receta, un modelo (no catálogo), una fábrica, una guia (no espiritual), una estructura.
- La vamos a utilizar para crear objetos iguales pero diferentes (?)
- Podemos decir que una clase es una colección abstracta o conceptual (no una lista o un arreglo) de objetos. Un paraguas que los envuelve.

Componentes de una clase



¿Que es un objeto?

- Es una instancia de una clase previamente definida. No puede haber objetos sin una clase que lo defina. Sin embargo, una clase sin objetos no sirve para nada
- Es como una clase se vuelve concreta y se puede utilizar.
- Un objeto a nivel conceptual puede ser cualquier cosa. Literalmente. Siempre y cuando podamos emitir un concepto sobre el mismo. Agrupa características y comportamientos similares.

Pensemos ejemplos...

Estado de los objetos

Es el valor que tienen los atributos de los objetos.

Objeto Lapiz001 Color: Azul

Objeto Lapiz002 Color: Verde

Objeto Lapiz003 Color: Rojo

Relación entre objetos

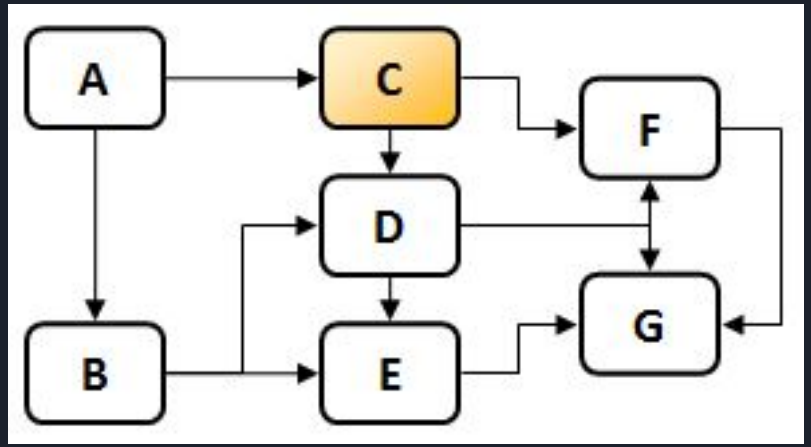
- **Asociación:** Se podría definir como el momento en que dos objetos se unen para trabajar juntos y así, alcanzar una meta. Para validar la asociación utilizamos la frase “usa un...”
- **Agregación/Composición:** Es un tipo de relación dependiente en dónde un objeto más complejo es conformado por objetos más simples. En esta situación utilizamos la frase “Tiene un”
- **Herencia:** Facilita la creación de objetos a partir de otros ya existentes e implica que una subclase (hija) obtiene el comportamiento (métodos) y características (atributos) de una super clase (padre). Usamos la frase “Es hijo de..”

Objetivos de la POO - Cohesión

Cuando decimos que un componente tiene una **alta cohesión** hablamos de que todos los elementos dentro de él están estrechamente relacionados

Objetivos de la POO - Acoplamiento

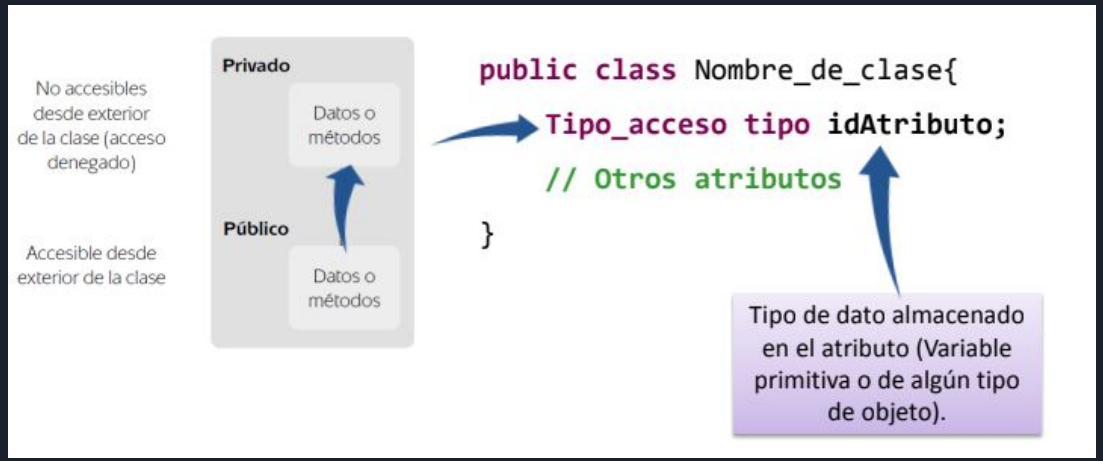
Cuando decimos que un componente tiene **bajo acoplamiento** hablamos del nivel de independencia que tiene un componente con respecto a los otros



Partes de una clase

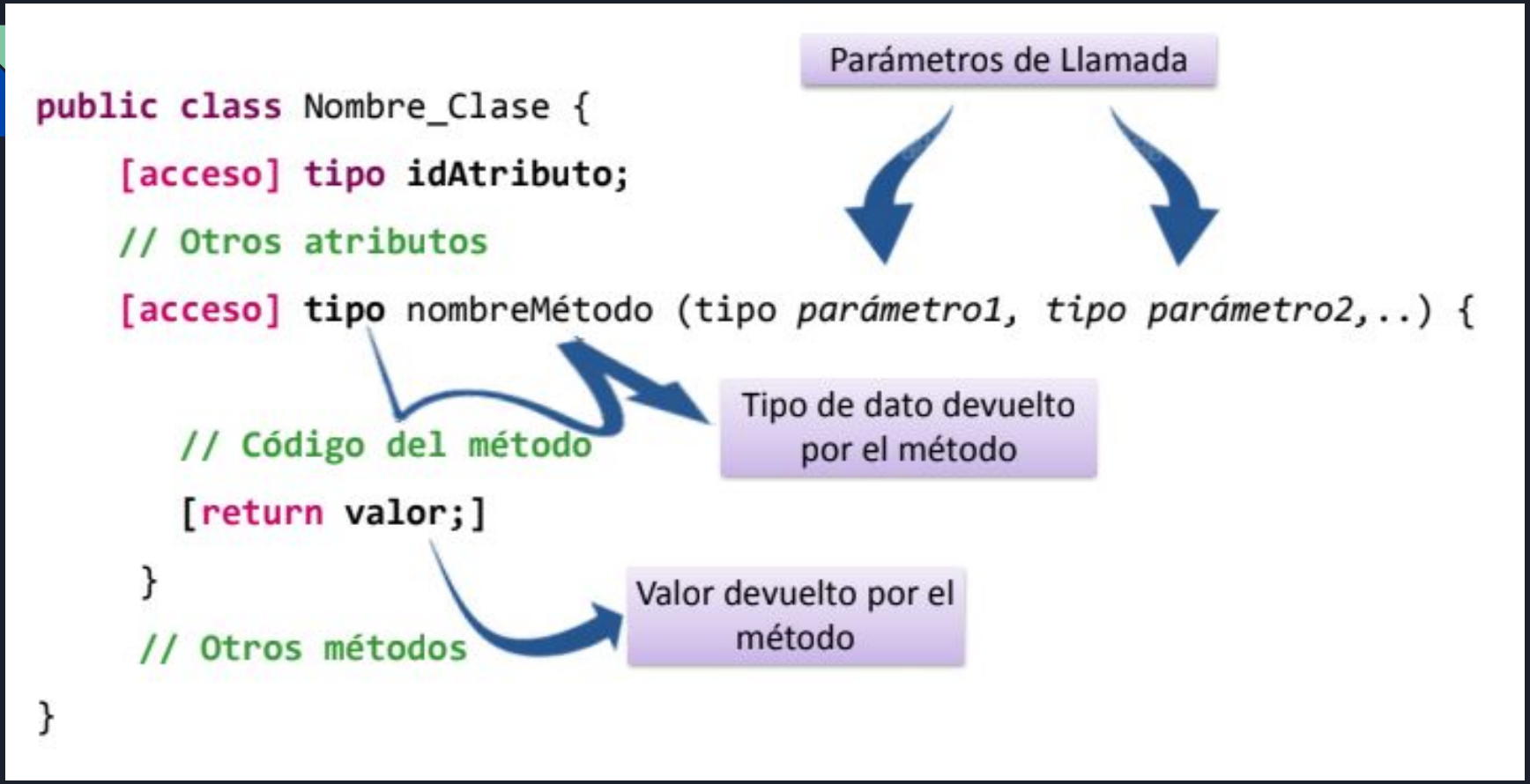
Atributo.

- Contenedor de un tipo de de datos asociado a un objeto (o a una clase de objetos), que hace los datos visibles o no desde fuera del objeto y esto se define como las características predeterminadas de un objeto y cuyo valor puede ser alterado por la ejecución de algún método.



Método.

- Los métodos definen el comportamiento de los objetos de una clase. Pueden almacenar, mostrar o realizar alguna operación con la información del objeto.
- Se accede a los métodos de un objeto a través del operador punto (.)
- Un método también es un bloque de código al que se le puede transferir el control de ejecución de un programa y ejecutar dicho código.
- Los métodos se suelen dividir en los que se usan internamente en la clase (private) y los que son accesibles desde el exterior (public)
- Los métodos en Java son siempre miembros de clases. Fuera de ellas, no hay.


Parámetros de Llamada
public class Nombre_Clase {
[acceso] tipo idAtributo;
// Otros atributos
[acceso] tipo nombreMétodo (tipo parámetro1, tipo parámetro2,..) {
// Código del método
[return valor;]
}
// Otros métodos
}
Tipo de dato devuelto por el método
Valor devuelto por el método

Constructores.

Constructores.

- Es una función o método que se llama automáticamente al crear un objeto de una clase. (al hacer new)
- El constructor reserva memoria para el objeto e inicializa sus variables.
- No tienen retorno (ni siquiera void) y se llaman igual que las clase.
- Se permiten varios constructores, cada uno con parámetros de entrada diferentes.
- Un constructor puede invocar otro constructor de la misma clase usando el operador this. Siempre en la primera línea.
- Java crea automáticamente un constructor vacío e inicializa las variables con su valor por defecto.

Constructores y Herencia.

- El constructor de una clase hija puede llamar al constructor de su clase padre mediante el operador: **super**
- Como norma universal, cada vez que en Java se crea un objeto de una clase, antes de ejecutarse el constructor de dicha clase se ejecutará primero el de su superclase.

Clases finales.

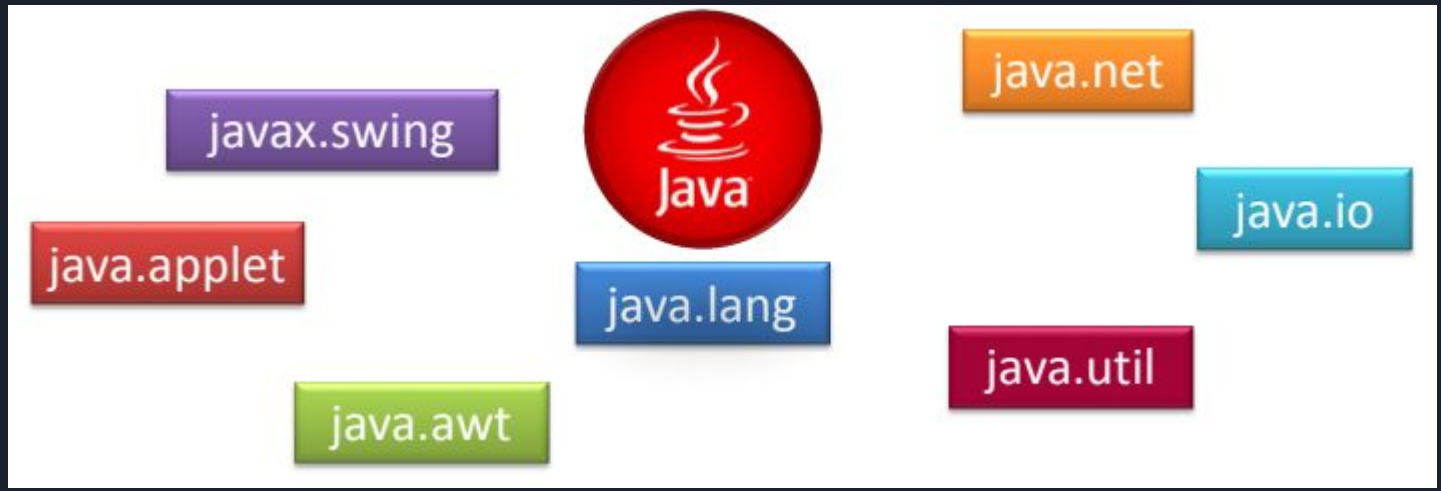
- Si queremos que una clase sea NO heredada por otra, deberá ser declarada con el modificador **final** luego de **public**
- Si otra clase intenta heredar de una clase final, se producirá un error que no nos permitirá la compilación

Sobrecarga de métodos.

- Cuando una clase hereda de otra, el comportamiento de los métodos que hereda no siempre se ajusta a las necesidades de la nueva clase. En estos casos, la subclase puede optar por volver a reescribir el método heredado, es lo que se conoce como sobrescritura de un método.
- Reglas a la hora de sobreescribir un método:
  - Cuando se sobrescribe un método de una subclase, éste debe tener el mismo formato que el método de la superclase que sobrescribe.(igual nombre, iguales parámetros e igual tipo de devolución).
  - El método sobrescrito puede tener un modificador menos restrictivo que el de la superclase. Por ejemplo, el método de la superclase puede ser protected y la versión sobrescrita en la subclase puede ser public, pero nunca uno más restrictivo.

Biblioteca de clases de Java.

- Java incorpora una amplia biblioteca de clases e interfaces denominado Java API; sus clases se pueden utilizar para formar otras nuevas, crear objetos, utilizar sus métodos.



Modificadores de Acceso.

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Private.

Su uso está restringido al interior de la clase. Lo que significa que solamente puede ser utilizado en el interior de su misma clase. Este modificador puede ser aplicado a métodos y atributos, pero no a la clase.

Ninguno, sin definir.

La no utilización de modificador proporciona lo que se conoce como el acceso por defecto. Si una clase, atributo o método tienen acceso por defecto, únicamente las clases de su mismo paquete tendrán acceso al mismo.

Protected.

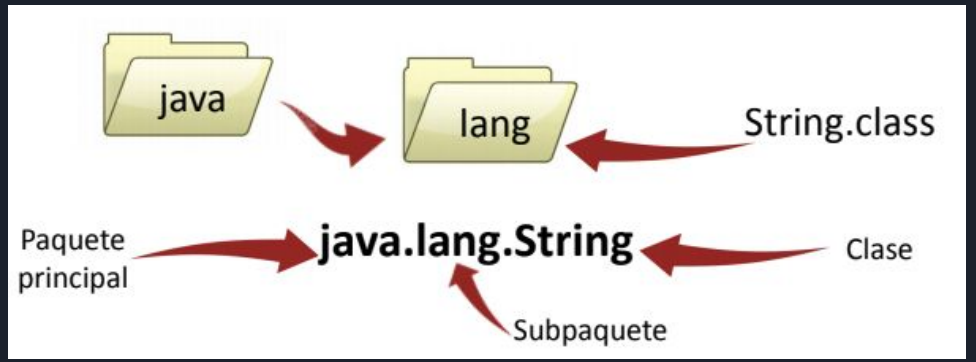
Es empleado en la herencia. Cualquier método o atributo definido como protected en una clase puede ser utilizado por cualquier otra clase de su mismo paquete y además por cualquier subclase de ella, independientemente del paquete en que esta se encuentre. Una clase no puede ser protected, sólo sus miembros

Public.

Este modificador ofrece el máximo nivel de visibilidad. Un elemento (clase, método o atributo) public será visible desde cualquier clase, independientemente del paquete en que se encuentre.

Organización de Clases: Los Paquetes.

- Los paquetes es la forma que tiene Java de organizar los archivos con las clases necesarias para construir las aplicaciones. Java incorpora varios de ellos, por ejemplo: java.lang, java.io, o java.util, con las clases básicas para la construcción de programas: System, String, Integer, Scanner.
- Un paquete es un directorio en el que se almacenan los archivos .class con los bytecodes de la clase; un paquete puede a su vez estar compuesto de otros subpaquetes.



Organización de Clases: Los Paquetes.

- Permiten organizar las clases de manera estructurada. Los paquetes permiten agrupar clases que tengan algún tipo de relación lógica. Esto facilita su localización y utilización en un programa.
- Evitan conflictos de nombres. Una clase localizada en un determinado paquete se identifica mediante lo que se conoce como nombre cualificado de la clase. Este se compone del nombre de la clase, precedido por los nombres de los subpaquetes hasta llegar al paquete principal, separados por un “.” Esto permite que en un programa se pueda utilizar dos o más clases que tengan el mismo nombre y que se encuentren en distintos paquetes.