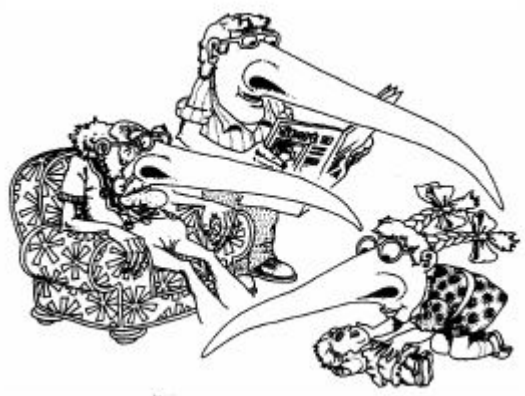
Programación Orientada a Objetos: Herencia

Tecnicatura Universitaria en Web

Herencia



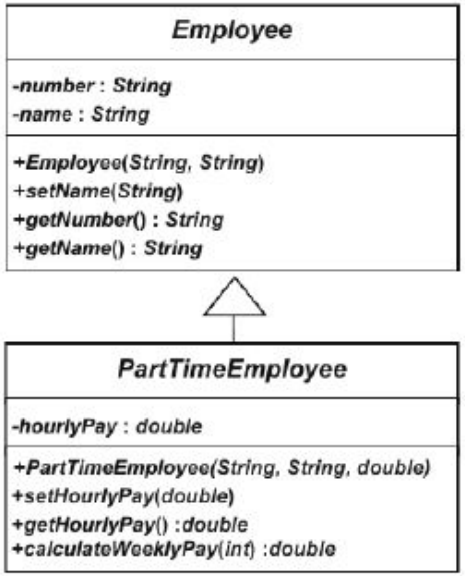
Es una relación entre las clases en la cual una clase comparte la estructura y el comportamiento de una (Herencia Simple) o más clases (Herencia Múltiple).

- ***La clase de la que las otras heredan se llama Superclase.***
- ***Análogamente la clase que hereda de otra/s clase/s se llama Subclase.***

Herencia

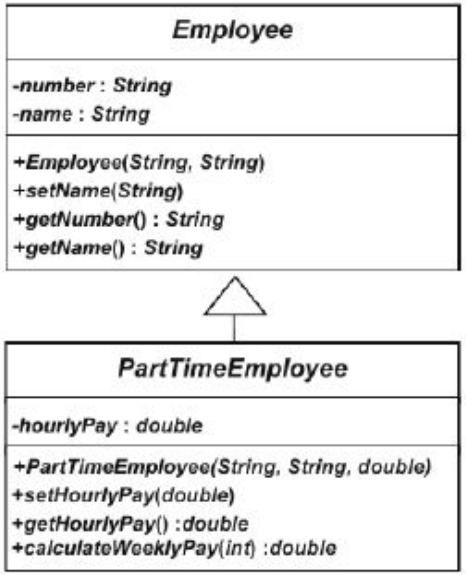
Con la Herencia una clase se puede definir en términos de otra. La nueva clase Hereda los atributos y métodos y además tiene atributos y métodos propios.

Herencia



- ***Employee*** es la clase base o la ***super clase*** o ***clase padre***.
- ***PartTimeEmployee*** es la ***subclase***, ***clase derivada*** o ***clase hijo***.
- Los atributos y métodos de ***Employee*** son heredados por ***PartTimeEmployee***.

Herencia



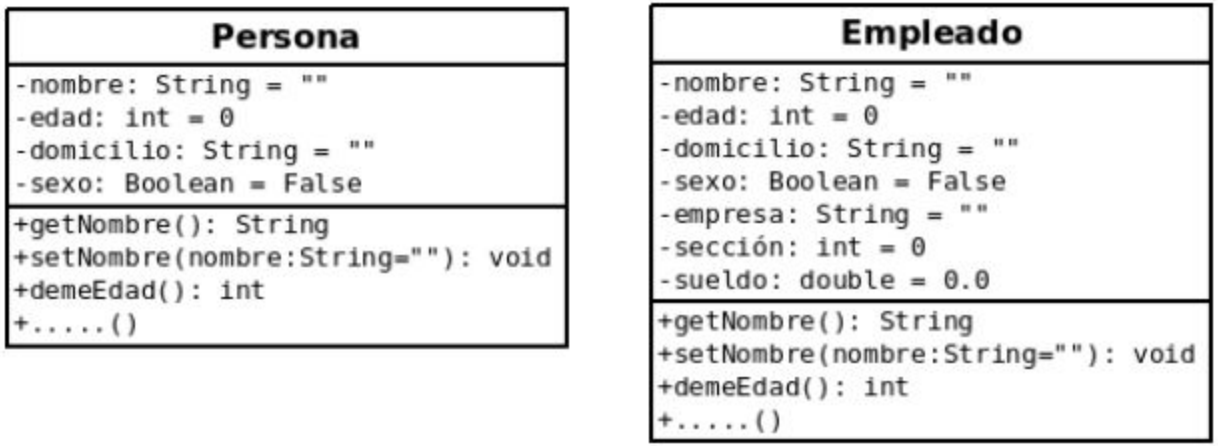
- La herencia es una relación jerárquica.
- La herencia también se referencia como una relación ***es una clase de***. Un ***PartTimeEmployee es una clase de Employee***.
- La clase ***PartTimeEmployee*** hereda los atributos y métodos de ***Employee*** y además tiene atributos y métodos propios.

Herencia

Personas

Empresario

Herencia

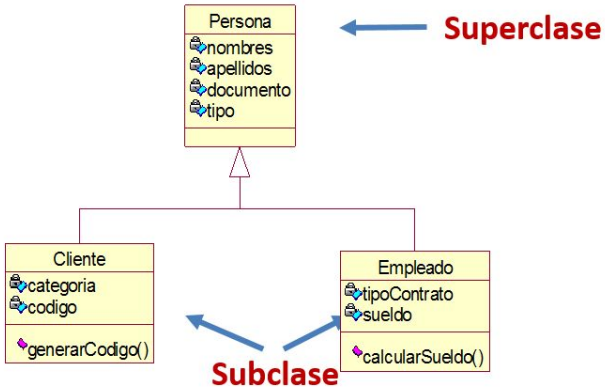


- Las dos clases son muy similares. Ambas clases tienen datos y muchas operaciones en común.
- Si se implementan las clases por separado sin observar las características en común entonces se duplica el código, es más difícil detectar errores, etc.

Herencia

Si se aprovechan las similitudes entonces lo que se necesita implementar son solamente las nuevas características.

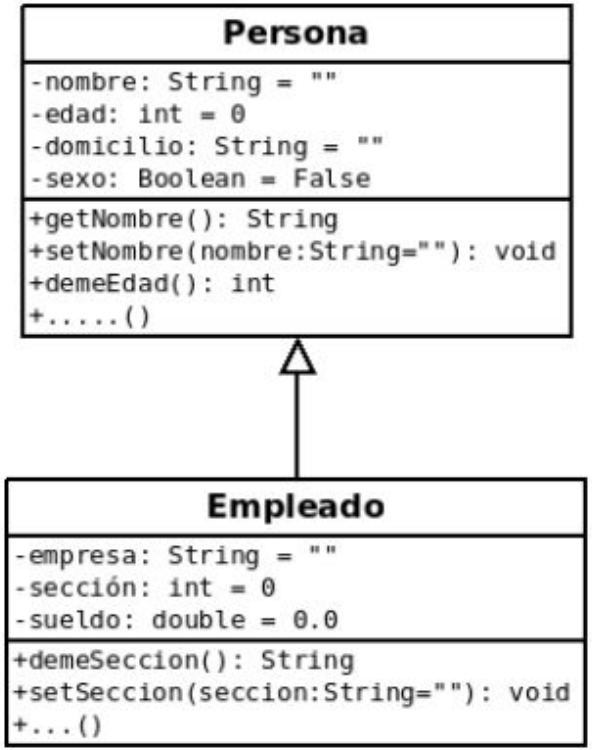
Herencia



Si B es una subclase de A entonces:

- B hereda todas las variables y métodos que no son privados.
- B puede definir nuevas variables y métodos de instancia propios.

Herencia

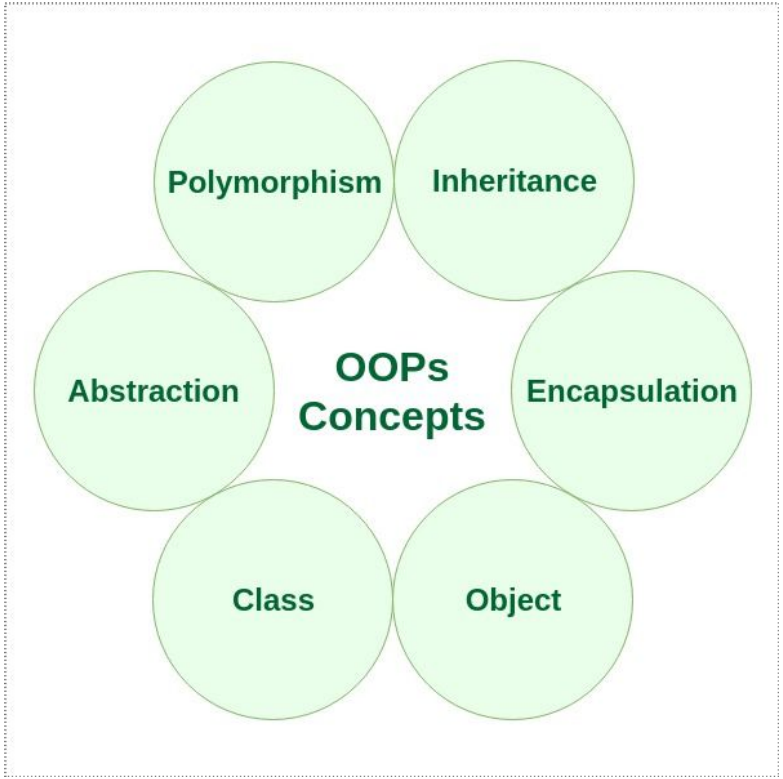


Una instancia de la clase Empleado hereda:

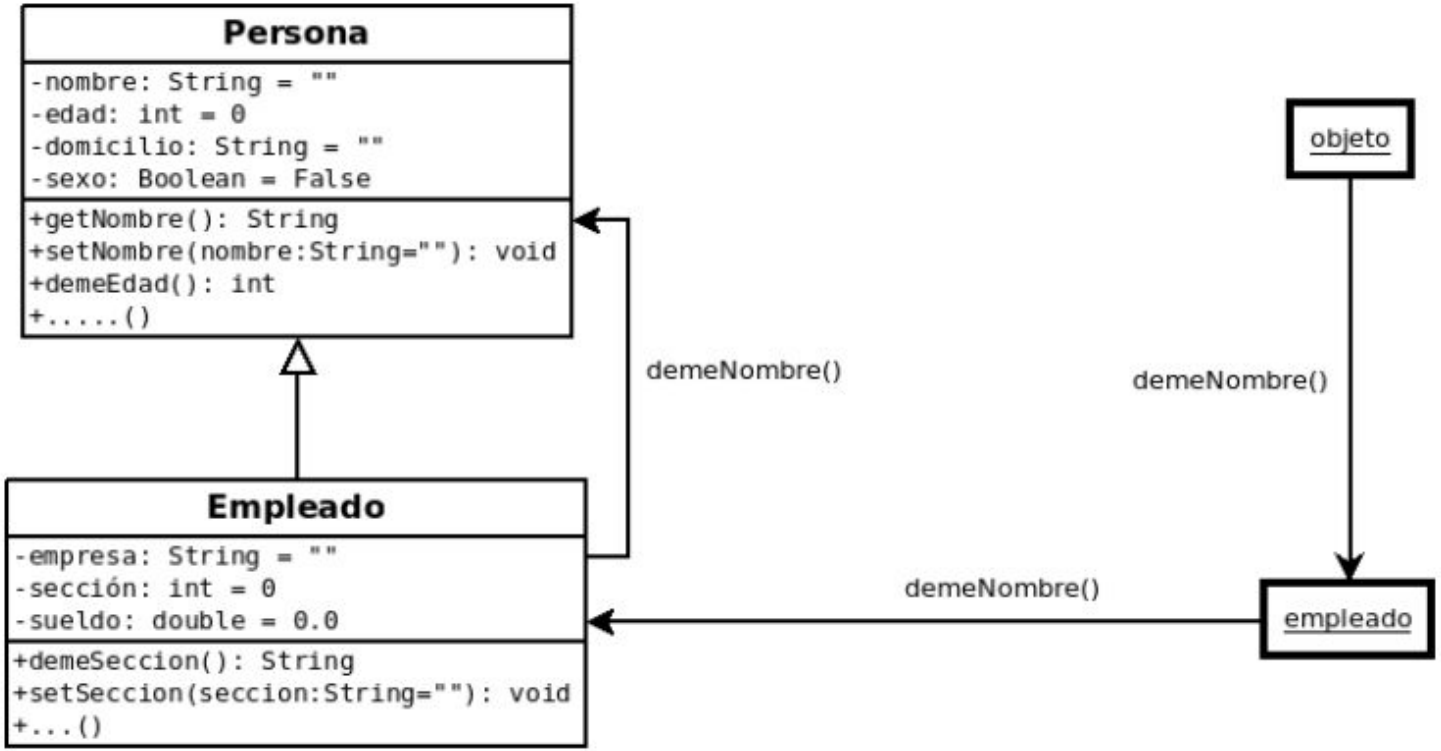
- Las variables de instancia: nombre, edad, domicilio, sexo.
- Los métodos: demeNombre(), demeDomicilio(), demeSexo(), ... etc.

Herencia

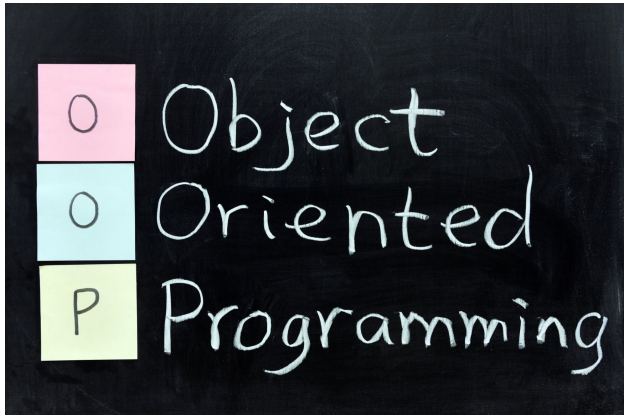
Cuando una instancia recibe un mensaje el método se busca en su clase si no está allí la búsqueda continúa en su superclase y así sucesivamente hasta llegar al tope. En este caso si el método no se encuentra se produce un error.



Herencia


Persona
-nombre: String = ""
-edad: int = 0
-domicilio: String = ""
-sexo: Boolean = False
+getNombre(): String
+setNombre(nombre:String=""): void
+demeEdad(): int
+.....()
objeto
demeNombre()
demeNombre()
Empleado
-empresa: String = ""
-sección: int = 0
-sueldo: double = 0.0
+demeSeccion(): String
+setSeccion(seccion:String=""): void
+...()
demeNombre()
empleado

Herencia en Java



1. Toda clase en Java es derivada de la clase Object.
2. Las clases en Java se organizan en jerarquías usando la palabra clave extends.
3. La relación de herencia establece una relación de superclase/subclase.
4. La superclase contiene los miembros comunes a sus subclases. Generalización.
5. La subclase contiene miembros específicos. Especialización.

Herencia en Java

c l a s s A ***extends*** B {

.....

}

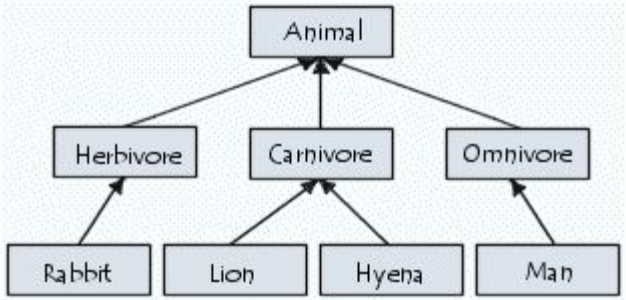
class Empleado ***extends*** Persona {

.....

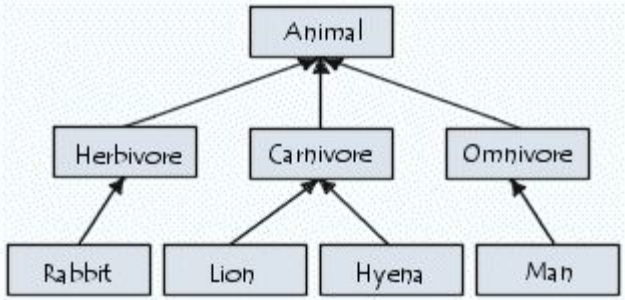
}

Herencia

- En el contexto de la Herencia es confuso pensar en el objeto producido por la clase derivada. Esto se debe a que hay dos clases en lugar de una.
- Desde el exterior se percibe que la nueva clase tiene la misma interfaz que la clase base y quizás algunos métodos/campos adicionales.
- Cuando se crea un objeto de la clase derivada, el mismo contiene un objeto de la clase base. Este subobjeto se encuentra envuelto dentro del objeto de la clase derivada.



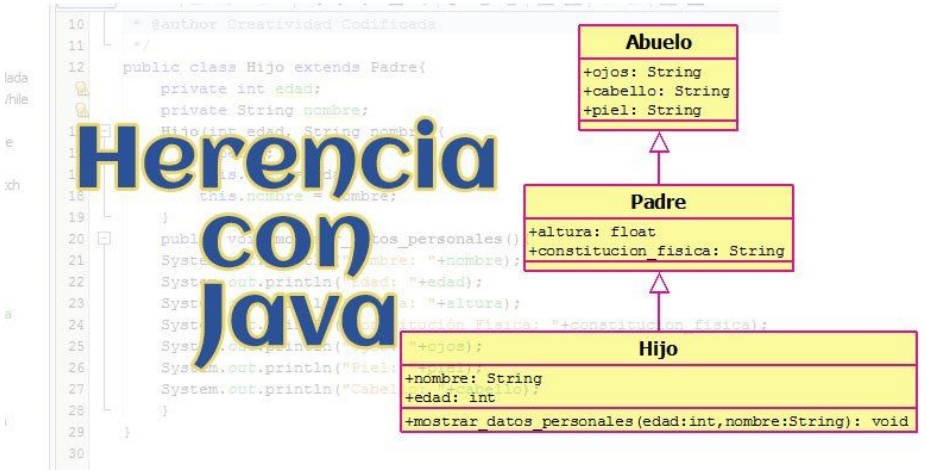
Herencia



La inicialización de los objetos creados mediante la Relación de Herencia se realiza en el constructor a través de la invocación del constructor de la clase base. Es importante notar que: Java inserta llamadas al constructor de la clase base en el constructor de la clase derivada.

Herencia

```
class Arte {
 public Arte () {
  System.out.println(”Constructor de Arte” ) ;
 }
}


class Dibujo extends Arte {
 Dibujo () {
  System.out.println (”Constructor de Dibujo” ) ;
 }
}

public class DAnimado extends Dibujo {
 public DAnimado ( ) {
  System.out.println(”Const. de DAnimado” ) ;
 }

 public static void main ( String [] args) {
  DAnimado x = new DAnimado ( ) ;
 }
}
```

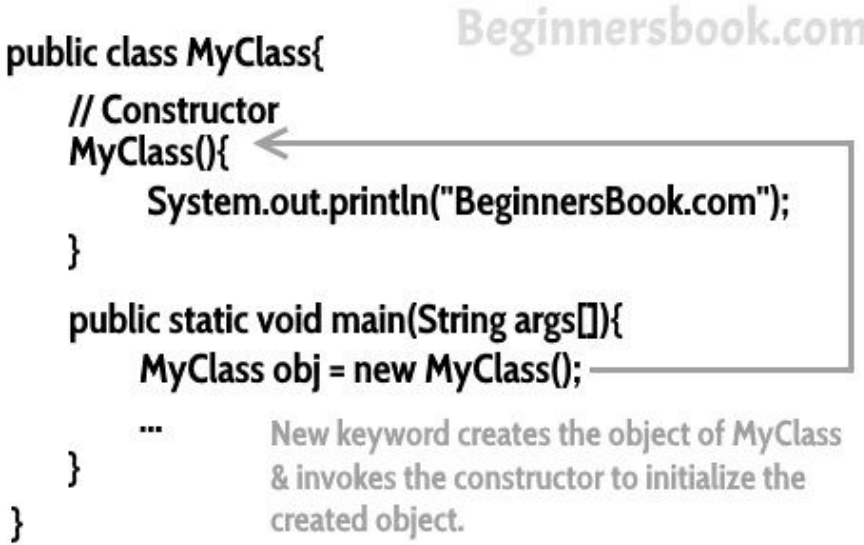


Herencia

Constructores con Argumento: en el ejemplo anterior los Constructores usados son los Constructores sin argumentos. No obstante, si se desea llamar al constructor de la clase base se debe invocar explícitamente a dicho constructor usando la palabra super y la lista de argumentos apropiados.

Herencia

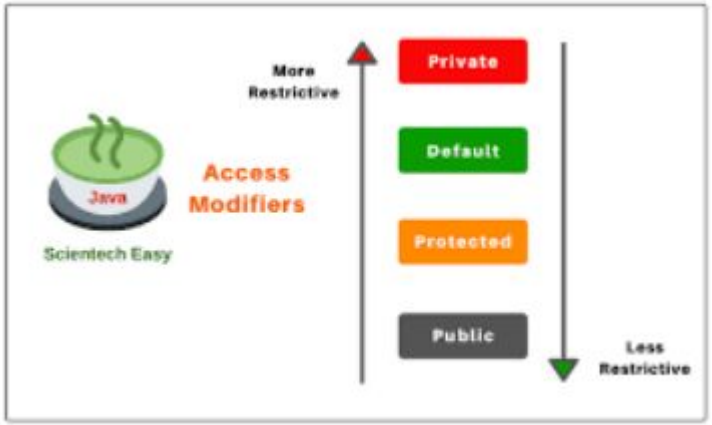
```
import static net.mindview.util.Print.* ;
class Game {
 Game(int i) {
  print("Game constructor") ;
 }
}

class BoardGame extends Game {
 Board Game (int i) {
  super ( i ) ;
  print("BoardGame constructor" ) ;
 }
}

public class Chess extends BoardGame {
 public Chess(){
  super ( 1 1 ) ;
  p r i n t ( " Chess c o n s t r u c t o r " ) ;
 }


 static void main(String [ ] args) {
  Chess x= new Chess();
 }
}
```



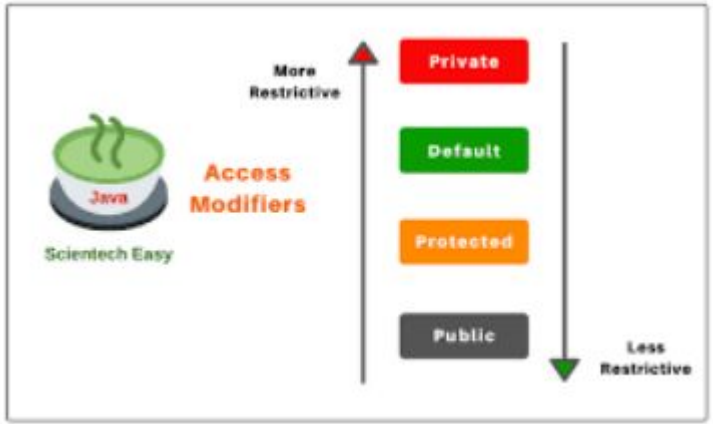
Herencia - Modificadores de Acceso

Se tiene dos niveles:

- A nivel de clases: ***public***, ***paquete***, ***private***.
- A nivel de miembros: ***public***, ***package***, ***protected***, ***private***.



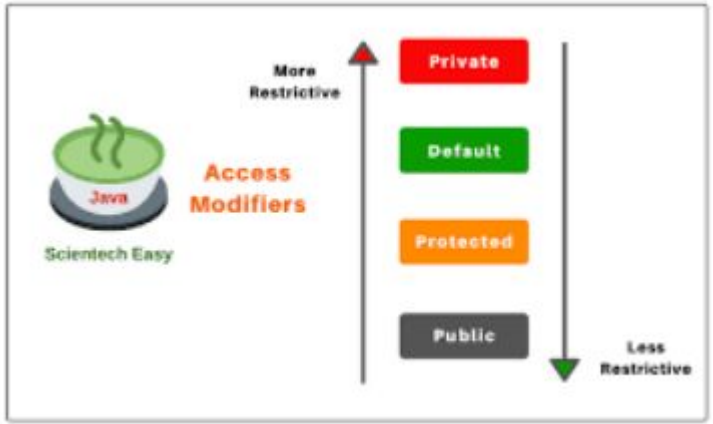
Herencia - Modificadores de Acceso



Nivel de Clase:

- ***Public:*** la clase es accesible para todas las clases en todas partes.
- ***Package:*** la clase es accesible sólo dentro de su paquete.
- ***Private:*** se utiliza para clases internas.

Herencia - Modificadores de Acceso



Nivel de Miembros:

- ***Public:*** el dato/método miembro es accesible para todas las clases en todas las partes.
- ***Protected:*** el dato/método miembro es accesible dentro de su propio paquete y en sus subclases.
- ***Package:*** el dato/método es accesible sólo dentro de su paquete.
- ***Private:*** el dato/método es accesible sólo dentro de su clase.