ENCAPSULAMIENTO Y MODIFICADORES DE ACCESO

- El encapsulamiento consiste en permitir la visibilidad de atributos y/o métodos.

- Hace uso de tres niveles de visibilidad:
  - Privados (private) se utilizan solo en esa clase.
  - protegidos (protected) se utilizan por todas los métodos, clases y /o atributos que se encuentre en el mismo package.
  - y los públicos (public) que pueden ser usados por cualquier clase o método.

ENCAPSULAMIENTO Y MODIFICADORES DE ACCESO

SIN ENCAPSULAR

```java
public class Profesor {

    public String nombre;
    public String identificacion;

    public void setNombre(String nomb) {
        nombre = nomb.toLowerCase();
    }
}
```

```java
public class Principal {
    public static void main(String args[]){
        Profesor objetoProfesor = new Profesor();
        objetoProfesor.setNombre("Jorge Robles");
        objetoProfesor.nombre="Felipe Montes";

    }
}
```

CON ENCAPSULADO

```java
public class Principal {
    public static void main(String args[]){
        Profesor objetoProfesor = new Profesor();
        objetoProfesor.setNombre("Jorge Robles");
        objetoProfesor.nombre="Felipe Montes";

    }
}
```

```java
public class Profesor {

    private String nombre;
    private String identificacion;

    public void setNombre(String nomb) {
        nombre = nomb.toLowerCase();
    }
}
```

ENCAPSULAMIENTO Y MODIFICADORES DE ACCESO

GETTERS Y SETTERS

- Los Setters y Getters son métodos de acceso lo que indica que son siempre declarados públicos, y nos sirven para dos cosas:
  - Setters: Nos sirve para asignar un valor a un atributo, nunca retorna nada (Siempre es void), y solo nos permite dar acceso público a ciertos atributos que deseemos el usuario pueda modificar.
  - Getters: Nos sirve para obtener (recuperar o acceder) el valor ya asignado a un atributo y utilizarlo para cierto método.

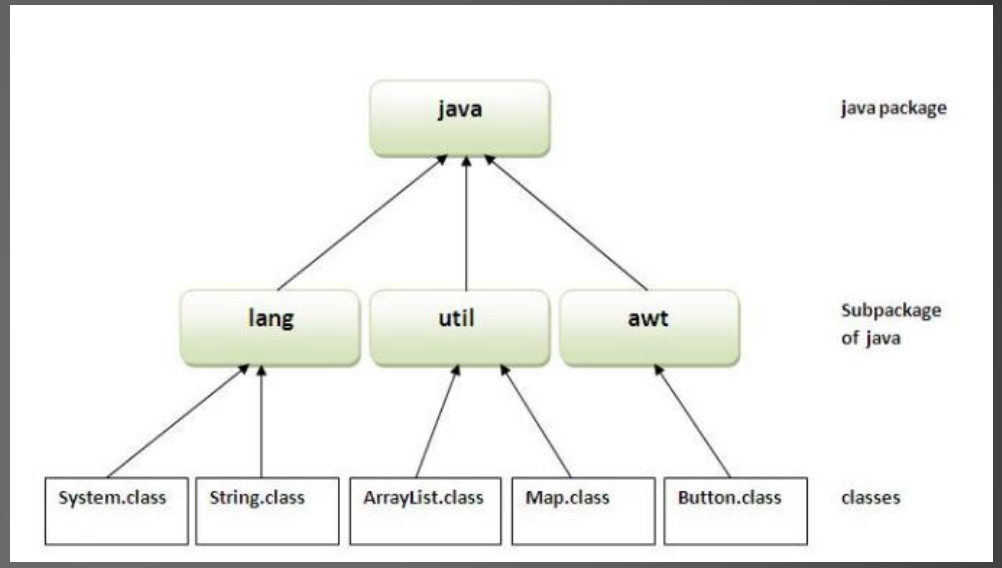
PAQUETES EN JAVA

- Un **Paquete** en Java es un contenedor de clases que por lo general tiene una funcionalidad y elementos comunes en un directorio de estructura jerárquica.

- VENTAJAS:
  - Agrupamiento de clases con características comunes.
  - Reutilización de código al promover principios de programación orientada a objetos.
  - Mayor seguridad al existir niveles de acceso.
  - Evita la colisión de clases que tengan el mismo nombre.
  - Mantenibilidad de código.
  - Brindan un nivel adicional de seguridad para nuestras clases, métodos o interfaces.

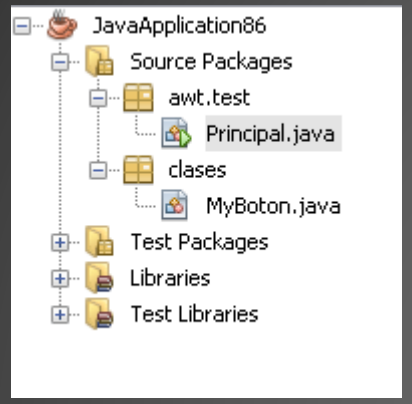
PAQUETES EN JAVA

- **CONSIDERACIONES:**
  - El paquete en Java se declara antes que cualquier otra cosa
  - Cada punto en la ruta del paquete es una nueva carpeta
  - Si no se declara un paquete (paquete por defecto)

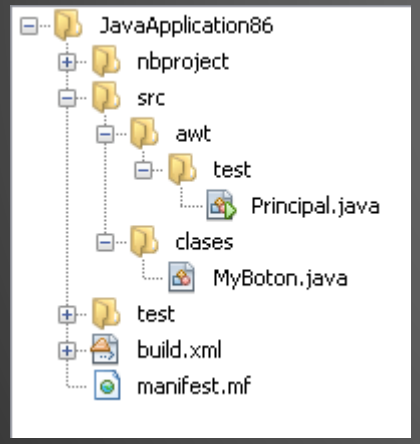



```
package otro_paquete.mi_paquete;
/*Se usa el punto para separar cada carpeta
  equivale a la ruta otro_paquete/mi_paquete dentro del proyecto*/
public class mi_clase
{

}
```

PAQUETES EN JAVA




```java
package awt.test;
import clases.*;
import clases.MyBoton;

public class Principal {
    public static void main(String args[]){
        MyBoton boton = new MyBoton();
    }
}
```




```java
package clases;
import java.awt.*;
public class MyBoton extends Button {

}
```

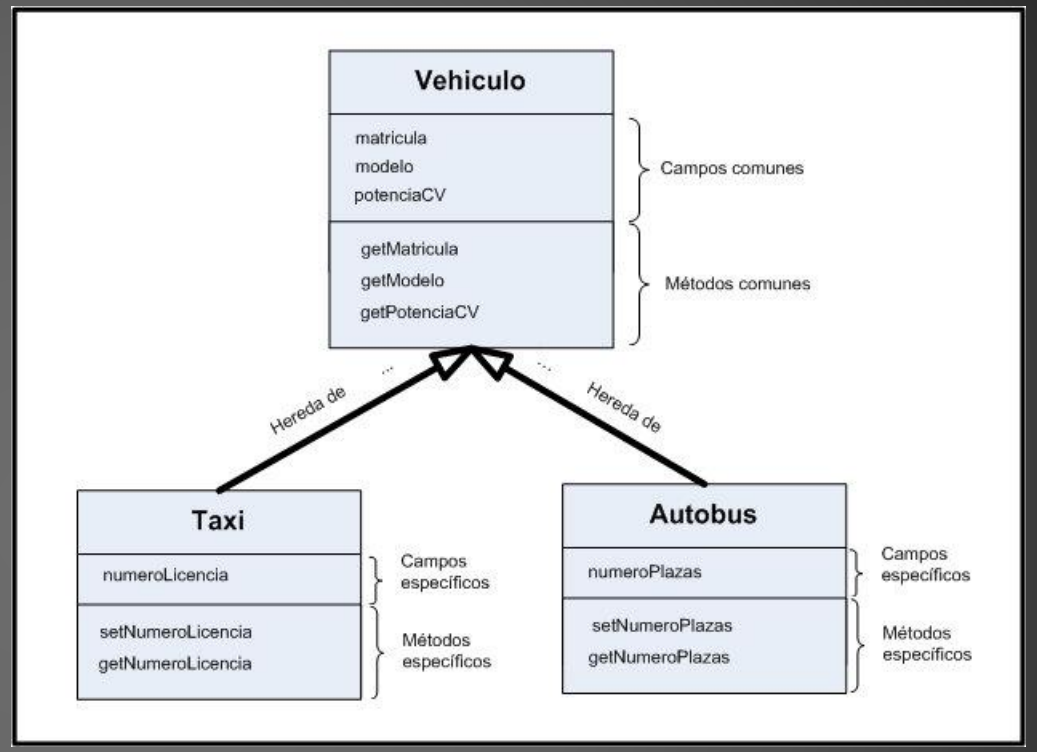
```java
package clases;
public class MyBoton extends java.awt.Button {

}
```

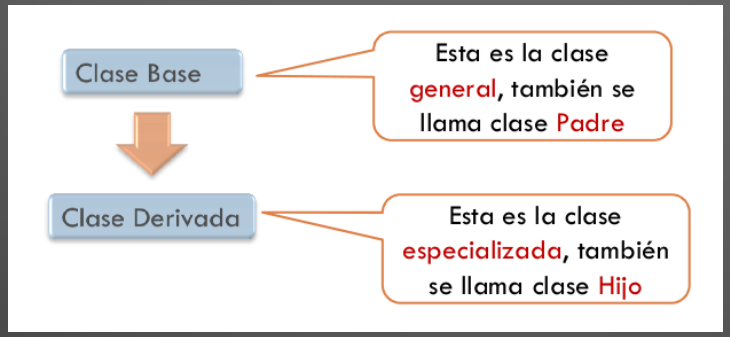
HERENCIA EN JAVA

- Permite definir una clase tomando como base a otra clase ya existente.
- Al heredar de una clase base heredaremos tanto los atributos como los métodos.
- Los constructores son utilizados, pero no heredados.
- Una ventaja es que nos permite evitar duplicado de atributos y de métodos.
- La herencia es un tipo de relación “ES UN….”.

HERENCIA EN JAVA

Taxi	Autobus
matricula modelo potenciaCV numeroLicencia	matricula modelo potenciaCV numeroPlazas
getMatricula getModelo getPotenciaCV getNumeroLicencia	getMatricula getModelo getPotenciaCV getNumeroPlazas





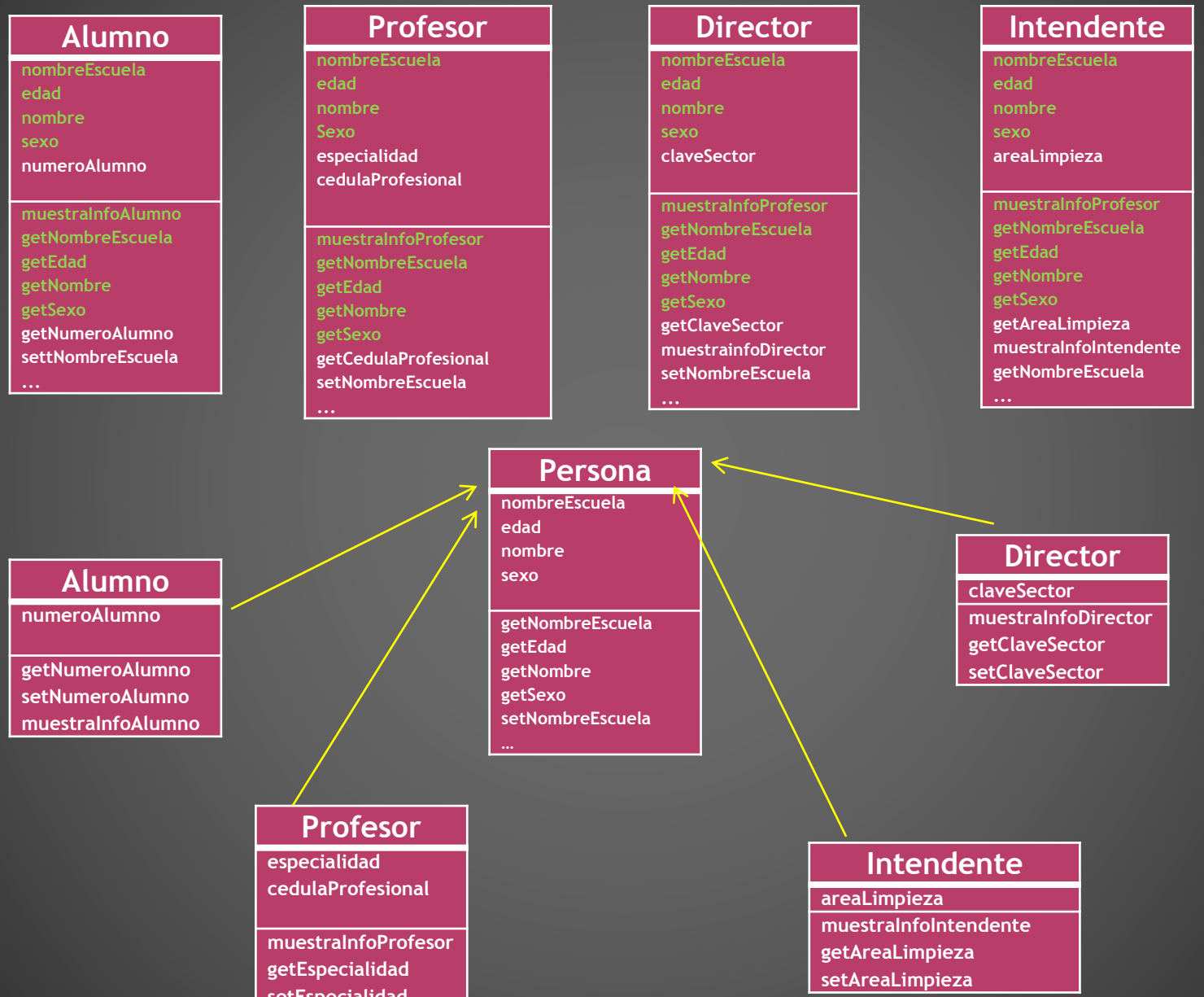
HERENCIA EN JAVA

SINTAXIS

```
<modificadores>  <nombre clase> [extends <superclase>]
{
        <atributos>
        <métodos>
}
```

```
public class Autobus extends Vehiculo {


}
```

HERENCIA EN JAVA



CONSTRUCTORES Y LA PALABRA SUPER

- EN LA HERENCIA LOS CONSTRUCTORES NO SON HEREDADOS
- CON LA PALABRA SUPER, MANDAMOS LLAMAR AL CONSTRUCTOR PADRE
- AL USAR LA PALABRA SUPER, DEBE SER LA PRIMER LINEA EN EL CONSTRUCTOR DE LA SUBCLASE
- SIEMPRE SE EJECUTA PRIMERO EL CONSTRUCTOR DE LA CLASE PADRE, DESPUES LAS SUBCLASES

```java
public class Persona {
    private static String nombreEscuela;
    private int edad;
    private String nombre;
    private String sexo;

    public Persona(int edad, String nombre, String sexo){
        this.edad=edad;
        this.nombre=nombre;
        this.sexo=sexo;
    }
```

```java
public class Alumno extends Persona{
   private int numeroAlumno;

    public Alumno(int numeroAlumno, String nombre, int edad, String sexo){
        super(edad,nombre,sexo);
        this.setNumeroAlumno(numeroAlumno);
    }
```