educación IT
CODE YOUR FUTURE

Java Standard Web Programming

Módulo 2

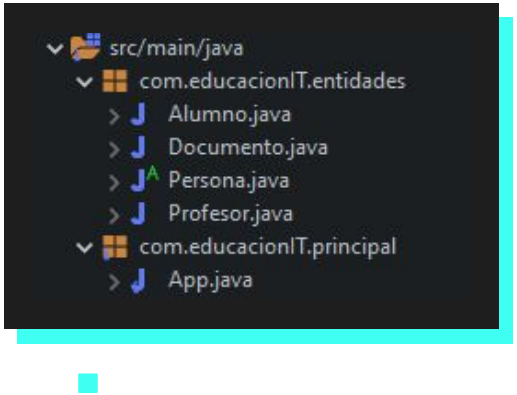
educación IT
CODE YOUR FUTURE

Paquetes y modificadores de acceso

Paquetes

Son **carpetas que se encuentran dentro del proyecto Java** que ayudan a organizar y contener un conjunto de **clases relacionadas por finalidad, ámbito o herencia**.

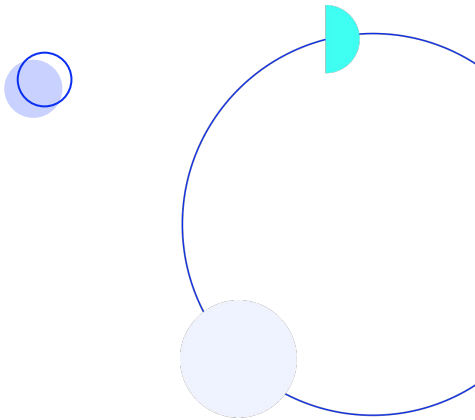
Los paquetes resuelven el problema del conflicto entre los nombres de las clases. Al crecer el número de clases hay probabilidad de designar con el mismo nombre a dos clases diferentes. Además, las clases, atributos y métodos tienen ciertos privilegios de acceso y los paquetes ayudan con esto.



Modificadores de acceso

Permiten dar un nivel mayor de seguridad a las aplicaciones. **Restringen el acceso a diferentes clases, atributos, métodos, constructores.** Aseguran que el usuario siga la "*ruta*" que especificó el desarrollador para acceder a la información.

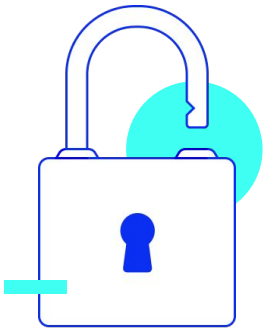
Es muy posible que nuestras aplicaciones sean utilizadas por otros programadores o usuarios con cierto nivel de experiencia. Con los **modificadores de acceso** nos aseguramos de que **no se modifiquen valores en forma incorrecta**. Se colocan primero, al crear una clase, atributo o método.



Default

Java nos da la opción de **no usar un modificador de acceso**. Al no hacerlo, el elemento tendrá un acceso conocido como **`default`** o *acceso por defecto* que permite que, tanto la **propia clase como las clases que se encuentren en el mismo paquete, accedan a dichos componentes** (de aquí la importancia de declararle siempre un paquete a nuestras clases).

Las clases, atributos, métodos y constructores pueden tener este tipo de acceso.



Ejemplo

```
class Auto {
    String color;
    String marca;
    String patente;
    boolean encendido;

    // Constructor
    Auto(){

    }

    // devuelve una cadena de caracteres con las caracteristicas que posea el objeto
    String mostrarDatos() {
        String mensaje = "El Auto es de color " + color + ", marca " + marca
                + ", patente " + patente + " y se encuentra "
                + ((encendido) ? "encendido" : "apagado");

        // la palabra reservada return le indica al metodo que finalizo su ejecuion
        // y que devuelva el objeto mensaje
        return mensaje;
    }

    //cambia el estado del atributo encendido sin devolver ningun dato
    void cambiarEstado(boolean encendido){
        this.encendido = encendido;
    }
}
```

Private

Es el modificador de acceso más *restrictivo*. **Cualquier elemento de una clase que sea privado puede ser accedido únicamente por la misma clase**.

Este modificador **no puede ser asignado a las clases**, ya que no tiene sentido crear una para que nadie pueda acceder a ella.

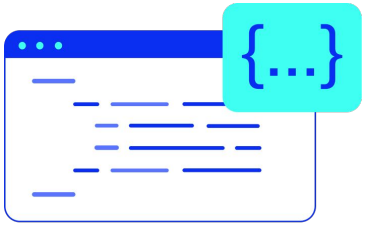
Ejemplo

```
class Auto {
    private String color;
    private String marca;
    private String patente;
    private boolean encendido;

    // Constructor
    private Auto(){

    }

    // devuelve una cadena de caracteres con las caracteristicas que posea el objeto
    private String mostrarDatos() {
        String mensaje = "El Auto es de color " + color + ", marca " + marca
                + ", patente " + patente + " y se encuentra "
                + ((encendido) ? "encendido" : "apagado");

        // la palabra reservada return le indica al metodo que finalizo su ejecuion
        // y que devuelva el objeto mensaje
        return mensaje;
    }

    //cambia el estado del atributo encendido sin devolver ningun dato
    private void cambiarEstado(boolean encendido){
        this.encendido = encendido;
    }
}
```

Protected

Permite acceder a los componentes solo desde la **misma clase**, **clases del mismo paquete** y **clases que hereden de ella** (incluso en diferentes paquetes).

Como nos pasa con el modificador de acceso `private`, **una clase no puede estar `protected`**.



Ejemplo

```
class Auto {
    protected String color;
    protected String marca;
    protected String patente;
    protected boolean encendido;

    // Constructor
    protected Auto(){

    }

    // devuelve una cadena de caracteres con las caracteristicas que posea el objeto
    protected String mostrarDatos() {
        String mensaje = "El Auto es de color " + color + ", marca " + marca
                + ", patente " + patente + " y se encuentra "
                + ((encendido) ? "encendido" : "apagado");

        // la palabra reservada return le indica al metodo que finalizo su ejecuion
        // y que devuelva el objeto mensaje
        return mensaje;
    }

    //cambia el estado del atributo encendido sin devolver ningun dato
    protected void cambiarEstado(boolean encendido){
        this.encendido = encendido;
    }
}
```

Public

Es el modificador de acceso más *permisivo*; **`public`** es lo contrario a **`private`** en todos los aspectos (lógicamente). Quiere decir que si un componente de una clase es `public`, tendremos **acceso a él desde cualquier clase o instancia** sin importar el paquete o procedencia.

Las clases, atributos, métodos y constructores pueden tener este tipo de acceso.

Ejemplo

```java
public class Auto {
    public String color;
    public String marca;
    public String patente;
    public boolean encendido;

    // Constructor
    public Auto(){

    }

    // devuelve una cadena de caracteres con las caracteristicas que posea el objeto
    public String mostrarDatos() {
        String mensaje = "El Auto es de color " + color + ", marca " + marca
                + ", patente " + patente + " y se encuentra "
                + ((encendido) ? "encendido" : "apagado");

        // la palabra reservada return le indica al metodo que finalizo su ejecuion
        // y que devuelva el objeto mensaje
        return mensaje;
    }

    //cambia el estado del atributo encendido sin devolver ningun dato
    public void cambiarEstado(boolean encendido){
        this.encendido = encendido;
    }
}
```

Cuadro comparativo de acceso

Modificador	La misma clase	Mismo paquete	Subclase	Otro paquete
`private`	✓	✗	✗	✗
`default`	✓	✓	✗	✗
`protected`	✓	✓	✓	✗
`public`	✓	✓	✓	✓

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!