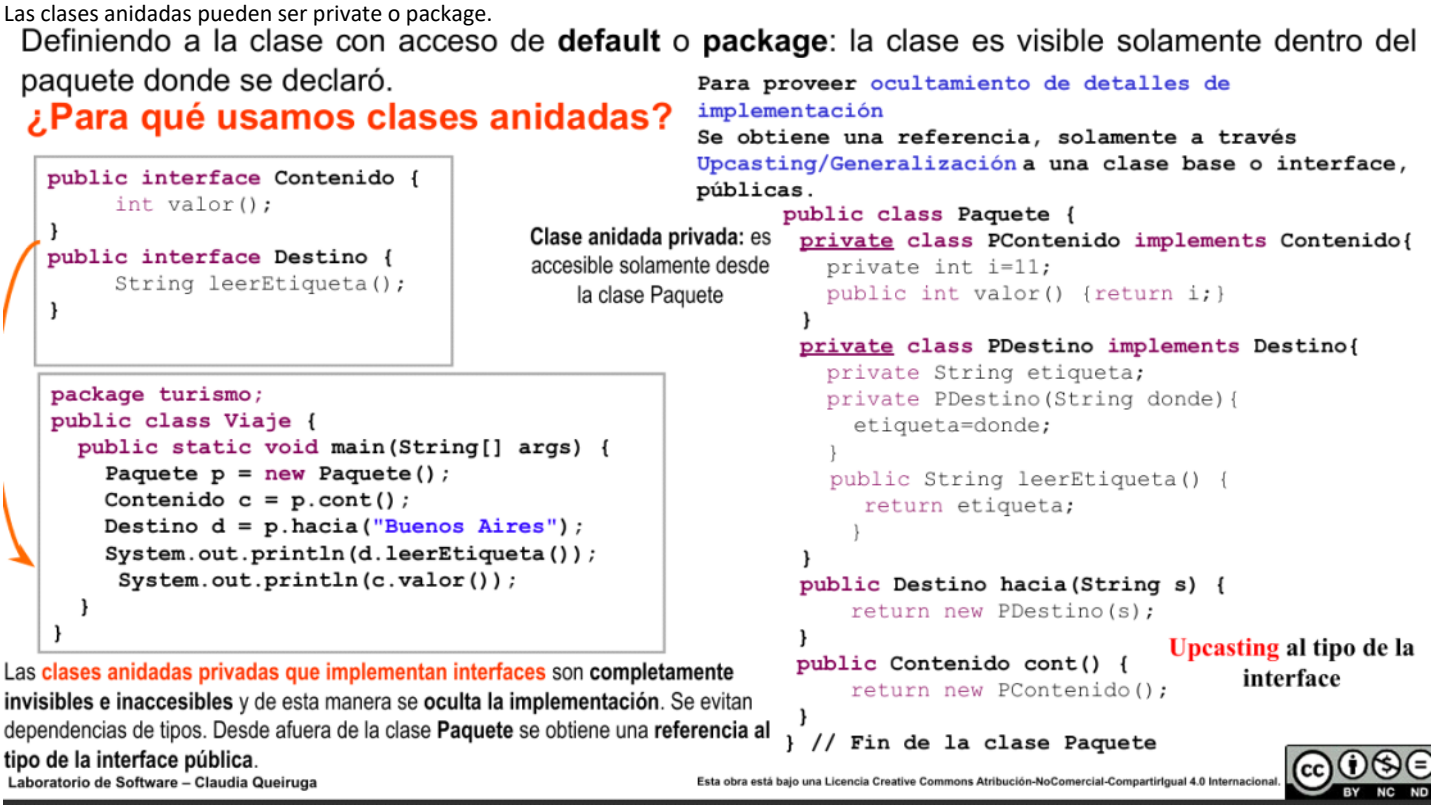
Clase 3-clases anidadas e internas

miércoles, 3 de septiembre de 2025 14:46

Clases adentro de clases, se usan cuando una clase es solo útil adentro de otra clase.
Si sirve en otro contexto, no tiene sentido y conviene sacarla afuera.
La clase anidada tiene acceso a todo el objeto que la contiene

La clase contenedora es quien hace el new de la clase.
La clase anidada es como una variable anidada.
Un objeto de una clase anidada tiene una referencia implícita al objeto de la clase que lo instanció (clase contenedora). A través de dicha referencia tiene acceso al estado completo del objeto contenedor, inclusive a sus datos privados. Por lo tanto las clases anidadas tienen más privilegios de acceso que las de nivel superior.

Las anidadas me permiten ocultar aún jmás cosas de implementación.
Las clases anidadas pueden ser private o package.



es como que me permite tener las clases y la implementaci[on en privado y despúes tener interfaces públicas con las que otras clases usan lo que hice.

Clase local: clase andentro de método.
Es como una variable local. Vuelve a ser para ocultar implementación, se usa si solo un método usa algo.
Ej si la dedfino en un if solo puedo acceder ahí.

Clases anonimas: crear una clase sin nombre que implementa

Las **clases anónimas** son **clases locales** sin nombre. Se crean extendiendo una clase o implementando una interface. **Combinan** la sintaxis de **definición de clases** con la de **instanciación de objetos**. Las interfaces y los tipos enumerativos NO pueden definirse anónimamente.

El método **cont()** combina la **creación del valor de retorno** con la **definición de la clase que representa el valor retornado**.

- Crea un objeto de una clase anónima que implementa la interface **Contenido**.
- La referencia que devuelve el método **cont()** es automáticamente ***upcasteada*** a una referencia a Contenido.

```
public class Paquete {
  public Contenido cont() {
    return new Contenido(){
      private int i=11;
      public int valor(){
         return i;
      }
    };
  }
} // Fin de la clase Paquete
```

Es una **abreviatura** de la declaración de una clase que implementa la interface **Contenido**

```
public interface Contenido {
  int valor();
}
```

```
public Contenido cont() {

   class PContenido implements Contenido {
     private int i=11;
     public int valor(){ return i;}
   }

return new PContenido();
}
```

```
package turismo;
public class Viaje {
  public static void main(String[] args) {
    Paquete p = new Paquete();
    Contenido c = p.cont();
  }
}
```


📌 Ejemplo normal (clase con nombre)

Supongamos que tengo una interfaz:

java — Copy code

```java
public interface Contenido {
    int valor();
}
```

Una forma de implementarla sería:

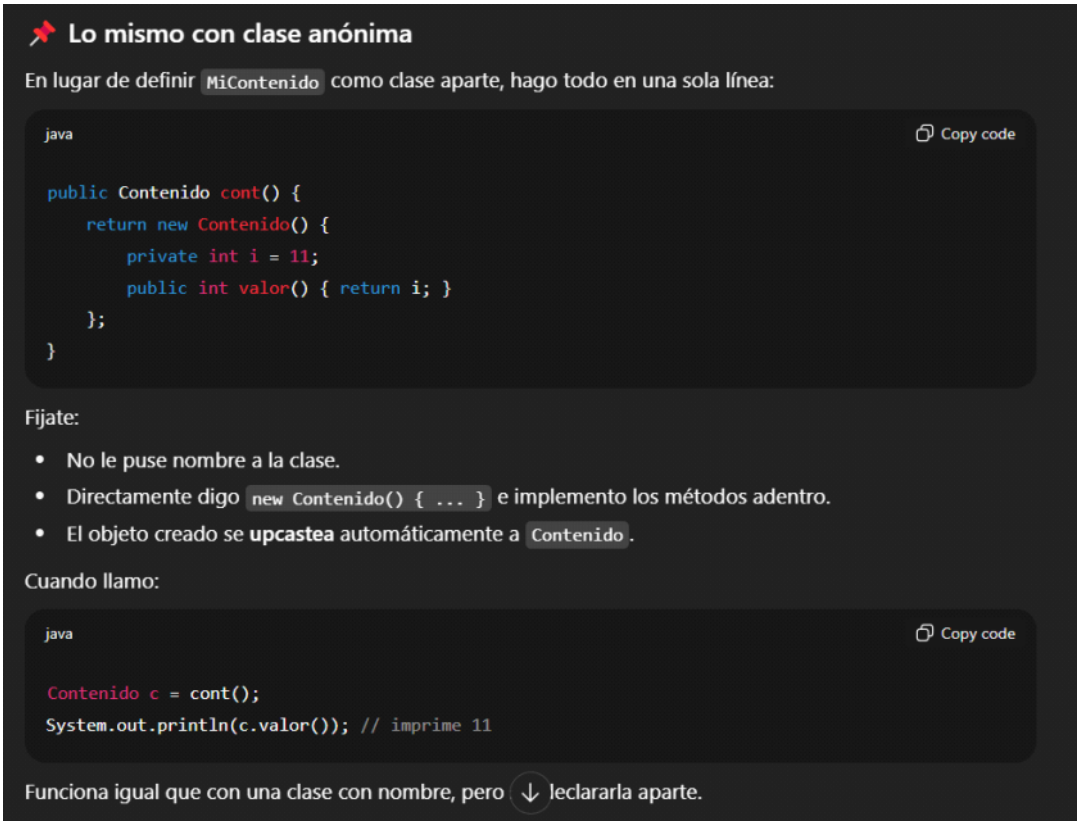
java — Copy code

```java
class MiContenido implements Contenido {
    private int i = 11;
    public int valor() { return i; }
}
```

Y la uso así:

java — Copy code

```java
Contenido c = new MiContenido();
```



No entendí nada de anónimas

Si a una clase anonima le paso como parámetro en un método una variable queda como final, no la puedo cambiar el valor
Las clases anónimas no tienen constructores pero puedo definir bloque de inicialización entre corchetes que se ejecutab cuando hago new.