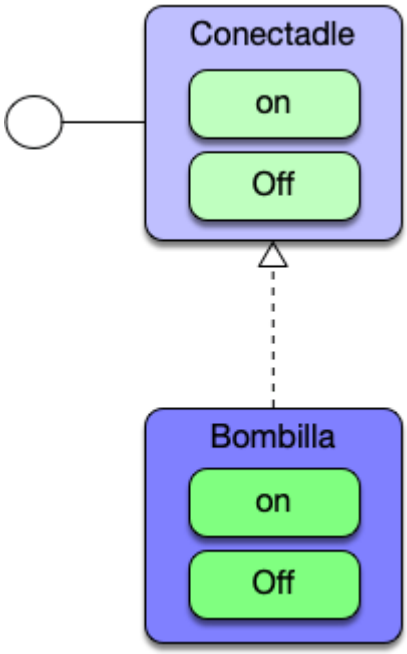
¿Kotlin Delegación y buenas prácticas?. El concepto de delegación a nivel de un lenguaje de programación es un concepto clásico y hace referencia a como nosotros podemos usar la composición para dividir responsabilidades entre varios objetos .Vamos a ver un ejemplo sencillo de delegación con Kotlin . En nuestro caso disponemos de una clase Bombilla que se puede encender y apagar . Esta clase implemente el interface Conectable.




```
interface Conectable {
    fun on()
    fun off()
}
```

Por lo tanto si queremos que la clase Bombilla implemente el interface nos será suficiente con construir una clase como esta:

```
class Bombilla:Conectable {

    override fun on() {
        println("la bombilla se enciende")
```

```
    }
    override fun off() {
        println("la bombilla se apaga")
    }
}
```

Hasta aquí todo correcto . Si ejecutamos este código en un programa main como el que sigue:

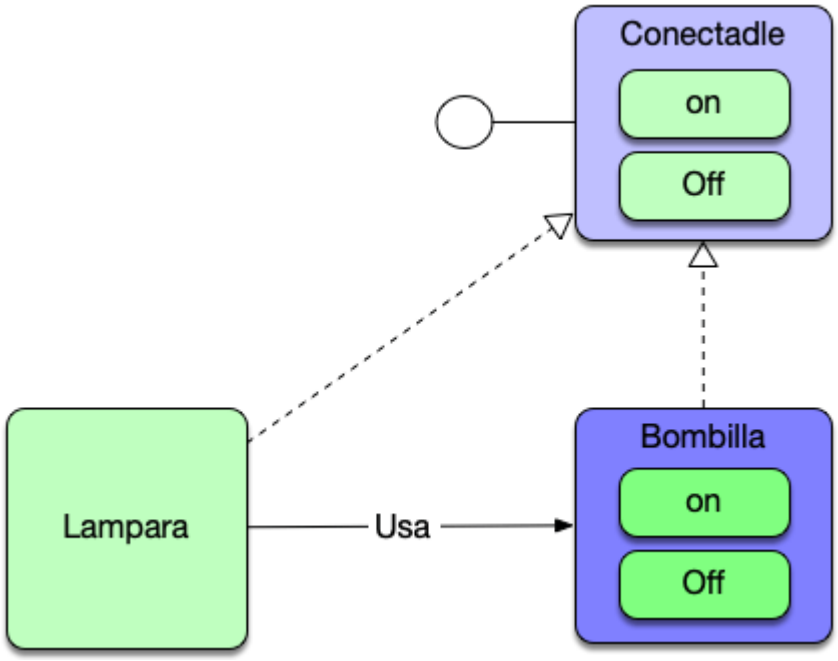
```
fun main(args: Array<String>) {
    var bombilla=Bombilla()
    bombilla.on()
    bombilla.off()

}
```

Nos imprimirá en consola que la Bombilla se encience o se apaga:

```
la bombilla se enciende
la bombilla se apaga
```

Kotlin Delegación

Todo correcto ahora bien ahora podemos construir la clase Lampara . Esta clase contiene una Bombilla e implementa el interface Conectable ya que una Lampara se puede encender y apagar. La peculiaridad que tiene la Lampara es que lo que enciende y apaga es su Bombilla por lo tanto podremos diseñar el código vía delegación y hacer que al encender la Lampara se encienda la Bombilla .



Vamos a verlo:

```
package com.arquitecturajava.pruebas

open class Lampara(var bombilla:Bombilla):Conectable {
    override fun on() {
       bombilla.on()
    }

    override fun off() {
        bombilla.off()
    }
}
```

Hasta aquí todo funciona correctamente y es un ejemplo clásico del concepto de delegación en donde una clase que usa composición delega en otra para implementar parte de su funcionalidad. Modificamos el programa main y volvemos a ejecutar:

```
fun main(args: Array<String>) {
```

```
    var lampara=Lampara(Bombilla())
    lampara.on()
    lampara.off()
}
```

El resultado es el mismo . Si nos fijamos la mejora que tiene este código sobre un código construido en Java es muy puntual . Ahora bien la clase Lampara en Kotlin se puede simplificar de la siguiente forma:

```
open class Lampara(var bombilla:Bombilla):Conectable by bombilla
```

Este código indica que la clase Lampara implementa el interface Conectable delegando en la Bombilla . La simplificación es total y es aquí donde un lenguaje como Kotlin brilla.

Otros artículos relacionados

- Java Herencia vs Interfaces
- Java interface default method y reutilización
- Java interface y extensibilidad