

## ENS IV – Tasca 03 part 2 (opcional)

Autor: Sergio Blanco Cuaresma

### **Enunciat**

Raonar si algun dels patrons de disseny que has vist es pot replantejar en la POA, justifica-ho i implementa un exemple bàsic.

### **Notes**

La pràctica ha sigut desenvolupada utilitzant Eclipse amb el plugin AJDT (<http://eclipse.org/ajdt/>) per fer ús d'AspectJ (<http://eclipse.org/aspectj/>).

### **Raonaments**

El patrons de disseny poden ser implementats utilitzant la programació orientada a objecte, i inclús millorar el disseny modular segons el paper “Design Pattern Implementation in Java and AspectJ” de Jan Hannemann i Gregor Kiczales:

<http://www.cs.ubc.ca/labs/spl/papers/2002/oopsla02-patterns.html>

<http://www.cs.ubc.ca/~jan/AODPs/>

A la web anterior es poden trobar exemples d'implementació de tots els patrons de GoF amb AspectJ:

- abstractFactory
- adapter
- bridge
- builder
- chainOfResponsibility
- command
- composite
- decorator
- facade
- factoryMethod
- flyweight
- interpreter
- iterator
- mediator
- memento
- observer
- prototype

- proxy
- singleton
- state
- strategy
- templateMethod
- visitor

## **Exemple: Singleton**

Veiem l'exemple concret de Singleton. Crearem 2 classes (ComptadorNonSingleton, ComptadorSingleton) que faran lo mateix:

```
package patterns;

public class ComptadorNonSingleton {
    private int comptador = 0;
    public void incrementarComptador() {
        comptador++;
    }

    public int getComptador() {
        return comptador;
    }
}
```

Però de les dos, voldrem que una tingi un comportament Singleton i l'altre no. Per fer això, en primer lloc utilitzarem un SingletonAspect generic:

```
public abstract aspect SingletonAspect {
    private Hashtable singletons = new Hashtable();

    public interface Singleton {}

    protected pointcut protectionExclusions();

    Object around(): call((Singleton+).new(..)) && !protectionExclusions() {
        Class singleton = thisJoinPoint.getSignature().getDeclaringType();
        if (singletons.get(singleton) == null) {
            singletons.put(singleton, proceed());
        }
        return singletons.get(singleton);
    }
}
```

Aquest s'encarrega de capturar les crides a la creació d'objectes del tipus indefinit “Singleton” i exclou els tipus retornats per “protectionExclusions()”.

A continuació farem el nostre Aspect específic anomenat “SingletonInstanceAspect”:

```
public aspect SingletonInstanceAspect extends SingletonAspect {
    declare parents: ComptadorSingleton implements Singleton;
    /*protected pointcut protectionExclusions():
        call((ComptadorNonSingleton+).new(..));*/
}
```

```
}
```

Aquest indica que el tipus generic “Singleton” correspondrà a la nostra classe “ComptadorSingleton”. Tenim la possibilitat de excloure altres objectes que heretin de “ComptadorSingleton”, però com que aquest no es el nostre cas, no cal indicar exclusions.

Finalment realitzarem el test de prova:

```
ComptadorNonSingleton c1[] = new ComptadorNonSingleton[10];
ComptadorSingleton c2[] = new ComptadorSingleton[10];

for (int i = 0; i < c1.length; i++) {
    c1[i] = new ComptadorNonSingleton();
    c1[i].incrementarComptador();
    c2[i] = new ComptadorSingleton();
    c2[i].incrementarComptador();
}

System.out.println("Comptadors NonSingleton");
System.out.println("-----");
for (int i = 0; i < c1.length; i++) {
    System.out.println(i + " >> valor comptador instancies: " + c1[i].getComptador());
}

System.out.println("Comptadors Singleton");
System.out.println("-----");
for (int i = 0; i < c2.length; i++) {
    System.out.println(i + " >> valor comptador instancies: " + c2[i].getComptador());
}
```

Es creen 10 objectes de cada tipus de comptador i en cada instanciació s'incrementa el seu comptador. Aquells que no siguin singleton tindran el comptador a 1 (valor inicial 0 + 1), mentre que els singleton tindran el comptador a 10, donat que s'haurà incrementat 10 vegades el mateix comptador.

Resultat:

```
Comptadors NonSingleton
-----
0 >> valor comptador instancies: 1
1 >> valor comptador instancies: 1
2 >> valor comptador instancies: 1
3 >> valor comptador instancies: 1
4 >> valor comptador instancies: 1
5 >> valor comptador instancies: 1
6 >> valor comptador instancies: 1
7 >> valor comptador instancies: 1
8 >> valor comptador instancies: 1
9 >> valor comptador instancies: 1
Comptadors Singleton
-----
0 >> valor comptador instancies: 10
1 >> valor comptador instancies: 10
2 >> valor comptador instancies: 10
3 >> valor comptador instancies: 10
4 >> valor comptador instancies: 10
5 >> valor comptador instancies: 10
```

```
6 >> valor comptador instancies: 10
7 >> valor comptador instancies: 10
8 >> valor comptador instancies: 10
9 >> valor comptador instancies: 10
```