# TP 3 – Programmation Orientée Aspects (POA)

## Objectifs pédagogiques

Comprendre et mettre en œuvre les grands principes de la POA ; utiliser l'outil AspectJ en Java.

## Introduction

Durant ce TP, vous utiliserez l'IDE Eclipse pour la programmation Java et son plugin AspectJ pour la programmation des aspects.

Installation d'AspectJ : avant de démarrer, vous allez installer le plugin AspectJ d'Eclipse sur les machines des salles de TP ou sur votre propre machine. Pour cela, reportez-vous au tutorial d'installation dans le forum « métier » du module Spiral.

## Prise en main d'AspectJ

1) Créez un projet AspectJ vide et importez-y l'application d'annuaire d'origine (celle disponible sur l'énoncé du TP 1)
2) Créez un nouvel aspect dans le projet, tant qu'à faire dans un package nommé « logging », avec le code en annexe.
3) Faites tourner ce programme et vérifiez la création et le remplissage du fichier log.txt dans le répertoire racine du projet.

## Modification d'un aspect existant

Reprenez le code de l'aspect ALog en suivant les étapes ci-dessous :

- Modifiez le pointcut « affichage » pour intercepter d'autres méthodes de sortie de la console System.out (print…)
- Modifiez l'advice correspondant pour afficher à l'écran la chaîne de caractères « [logged] », pour indiquer qu'un affichage a été loggé.
  Attention : vous allez devoir remodifier le pointcut pour ne pas qu'il intercepte les affichages de l'advice.
- Rajoutez un pointcut et un advice pour faire de même avec la console d'entrée (System.in).

## Conception d'un aspect

Dans cette partie, vous allez concevoir un nouvel aspect (que vous appellerez « APersist »), qui sera utilisé pour dissocier la gestion de la persistance du code métier de l'application. Plus précisément, il s'agit de modifier les classes de l'annuaire de façon à ce que la sauvegarde et le chargement de ce dernier se fasse *via* un aspect. Autrement dit toute référence à la sauvegarde et au chargement de l'annuaire doit disparaître du code source de ces classes.

Pour cela, vous suivrez la méthode suivante :

1) Passer en revue les éléments de votre application, et déterminer les classes métier et celles qui relèvent du traitement de la persistance. Séparer ces deux préoccupations dans des packages différents.

2) Repérer ensuite les points de jonction correspondant aux échanges de messages entre objets matérialisant le passage de l'une à l'autre des préoccupations (métier ←→ persistance). Formaliser les points de coupe permettant de les intercepter et supprimer les instructions correspondantes du code métier.
3) Mettre en place des code advices qui appellent les fonctionnalités de persistance désirées.
4) Repérer les éventuelles responsabilités « mixtes » entre des classes. Déplacer ce qui relève de la persistance dans l'aspect en utilisant des déclarations inter-types.

## Optimisation de l'annuaire

Vous allez maintenant modifier cet aspect pour :

- Ajouter une fonctionnalité permettant de modifier la description d'un site dont on donnera l'URL. En termes d'interface, une description vide reviendra à conserver l'ancienne description.
- Faire en sorte que la sauvegarde se fasse uniquement en cas de modification des données de l'annuaire (si la description est inchangée, ne pas faire de sauvegarde).

## Annexe : code de l'aspect ALog

```
package logging;

import java.io.FileWriter;

public aspect ALog {
	FileWriter logFile = null;

	/**
	 * Constructeur de l'aspect : initialise le fichier de log
	 */
	public ALog() {
		try {
			logFile = new FileWriter("log.txt", true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	/**
	 * Point de coupe qui intercepte les appels à la méthode println
	 * @param message le texte en paramètre de la méthode println
	 */
	pointcut affichage(Object message) : args(message) && call(void
java.io.PrintStream.println(*));

	/**
	 * Code advice contenant le code à exécuter lorsque l'aspect est
déclenché
	 * @param message le message à afficher
	 */
	after(Object message) : affichage(message) {
		try {
			logFile.write(new java.util.Date().toString() + " : "
+ message.toString());
			logFile.flush();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```