Introduction au couplage et à la cohésion

Pour justifier notre affirmation selon laquelle il existe de bonnes et de mauvaises conceptions, nous devons définir plusieurs termes qui nous permettront de discuter des aspects que nous considérons importants pour la conception de classes. Deux termes liés à la qualité de la conception de classes sont centraux : le *couplage* et la *cohésion*.

Le terme *couplage* fait référence à l'intensité des liaisons entre les classes. Nous avons déjà précisé dans les chapitres précédents que notre objectif est de concevoir des

applications composées d'un ensemble de classes qui communiquent par l'intermédiaire d'interfaces bien définies. Le degré de couplage indique la force de liaison entre les classes. Nous cherchons à réduire le degré de couplage, c'est-à-dire à obtenir un *couplage faible*.

Concept

Le terme **couplage** désigne l'importance des liaisons entre les classes. Nous nous efforçons de réduire le couplage dans un système – c'est-à-dire un système où chaque classe est largement indépendante et communique avec les autres classes par l'intermédiaire d'une interface réduite et bien définie.

Le degré de couplage détermine le niveau de difficulté à apporter des modifications dans une application. Dans une structure de classe fortement couplée, modifier une classe oblige à modifier plusieurs autres classes. C'est pourquoi nous tentons de l'éviter, car les conséquences d'une petite modification peuvent rapidement amener à modifier l'application tout entière. En outre, trouver tous les endroits où des modifications sont nécessaires et les réaliser toutes peut être difficile et long.

Dans un système faiblement couplé, nous pouvons souvent changer une classe sans modifier les autres, l'application continuera à fonctionner. Nous présenterons quelques exemples de couplages forts ou faibles dans ce chapitre.

Le terme *cohésion* reflète le nombre et la diversité des tâches confiées à une seule partie de l'application. La notion de cohésion est pertinente pour une classe ou une méthode[1].

Concept

La **cohésion** mesure la qualité du recouvrement entre une partie de code et la tâche logique à remplir. Dans un système très cohérent, chaque partie de code (méthode, classe ou module) est responsable d'une tâche bien identifiée. Une bonne conception de classes présente un haut degré de cohésion.

Idéalement, une partie de code devrait être en charge d'une tâche cohérente (c'est-à-dire une tâche qui peut être appréhendée comme une unité logique). Une méthode devrait implanter une opération logique et une classe devrait représenter une entité cohérente. La principale raison du principe de cohésion est la réutilisation : si une méthode ou une classe est responsable d'une seule opération bien définie, il est alors plus plausible qu'elle puisse être réutilisée dans un contexte différent. Quand la modification de certains aspects d'une application devient nécessaire, nous avons toutes les chances de trouver toutes les parties impliquées dans une même unité, ce qui constitue un autre avantage.

Nous allons étudier en quoi la cohésion influence la qualité de la conception des classes à l'aide des exemples suivants.

---

1. Nous utilisons parfois le terme de module (ou paquetage dans Java) pour faire référence à un groupe de plusieurs classes. La cohésion s'applique également à ce niveau.