1. Delegation von Aufgaben

Zusammengesetzte Objekte

Betrachten wir die Delegation anhand eines Beispiels. Das folgende Bild zeigt ein „Kreiseck“:


//Abb. 5.1: //Kreiseck mit Mittelpunkt im Ursprung des kartesischen Koordinatensystems

Unter einem „Kreiseck“ wird hier ein Quadrat – ein rechtwinkliges Viereck mit vier gleich langen Seiten – verstanden, welches von einem Kreis so ausgefüllt ist, dass die Seiten des Quadrats Tangenten an den Kreis sind. Mit anderen Worten, der Kreis soll einen Inkreis darstellen. Der Mittelpunkt des Kreisecks soll im Ursprung eines kartesischen Koordinatensystems liegen.

In Python kann ein solches „Kreiseck“ mit Hilfe einer Klasse erzeugt werden, die einen Kreis und ein Eck (Quadrat) aggregiert. Das Eck selbst stellt eine Aggregation von 4 Punkten dar. Sind die Klassen `Eck` und `Kreis` schon bekannt, so kann ein Objekt der Klasse `Kreiseck` aus Objekten der schon bekannten Klassen `Kreis` und `Eck` zusammengebaut werden. Siehe hierzu das folgende Bild:


//Abb. 5.2: //Delgation von Aufgaben

Methodenaufrufe einer Anwendung gehen an das zusammengesetzte Objekt der Klasse `Kreiseck`, z.B. `skaliere(2)`, was eine Vergrösserung um den Faktor 2 bedeuten soll. Die entsprechende Methode dieses Objektes leitet diese Botschaft dann weiter an das aggregierte Objekt der Klasse `Kreis` und das aggregierte Objekt der Klasse `Eck` durch Aufruf deren Skalierungsmethoden.

> Das „Gross“-Objekt delegiert den Aufruf, der an das „Gross“-Objekt selbst gerichtet war, weiter an die „Klein“-Objekte. Dieses Prinzip wird als **Delegationsprinzip** bezeichnet.

Alltagsbeispiele für Delegation

- Auto:
  - Der Motor delegiert die Fortbewegung an die Räder.
  - Der Fahrer delegiert die Zielfindung ans Navi (oder den Beifahrer/Co-Piloten).
- Restaurant:
  - Der Kellner delegiert das Zubereiten des Menüs an den Koch.
- Smartphone:
  - Der Benutzer delegiert das Auffinden von Informatioen via Browser an die Suchmaschine.
  - Das Gerät delegiert die Anzeigen der Suchresultate ans Display.
- Körper:
  - Das Gehirn delegiert die Bewegung des Arms via Nervenbahnen an die beteiligten Muskeln.

Einzelne Beteiligte einer Delegation reichen lediglich Signal an ihre Komponenten weiter ohne selber etwas hinzuzufügen. Die Verantwortung solcher „Durchlauferhitzer“ ist es, dafür zu sorgen, dass das Signal an der richtigen Stelle ankommt, welche in der Lage ist, die Operation auszuführen. Sie sorgen also für die **Integration** der **Operation** ins System. Dazu gibt es ein Prinzip, das hier kurz erwähnt werden soll.

Exkurs: Integration Operation Segregation Principle IOSP

Das IOSP ist eine Clean-Code-Strategie, die es uns ermöglicht, klar zwischen Integration und Operation zu unterscheiden:

- Reine **Integration** = Code, der anderen Code aufruft, selber jedoch keine Logik implementiert
- Reine **Operation** = Code, der Logik implementiert, dazu aber keinen anderen Code benutzt

Vorteil: Es ermöglicht uns beispielsweise, die Komplexität grosser Methoden systematisch auf viel einfachere zu reduzieren. Das Prinzip kann also helfen, Komplexität aufzubrechen und den Code klarer zu gestalten.

Achtung: Die Einteilung des Codes in reine Integration resp. Operation entspricht einem Ideal (Leitstern). Das IOSP kann und sollte daher nicht kopflos überall in der Praxis umgesetzt werden. Gerade aber für die Delegation von Aufgaben, macht es Sinn, sich beim Entwurf darüber Gedanken zu machen

Weitere Infos über das IOSP → Artikel von Ralf Westphal

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu97/theorie/lu5-kapitel_1?rev=1788091602

Last update: **2026/08/30 14:06**