

InClass-Exercise 13

hpaetzold@ethz.ch

Dezember 2025

1 Konstruktion (2022 W12)

In dieser Aufgabe implementieren Sie ein Verteilungssystem für Ressourcen eines Bauunternehmens, mit welchem Ressourcen (z.B. Sandsäcke, Kabelrollen, Ziegelpaletten) auf Baustellen verteilt werden. Das Interface **Resource** repräsentiert Ressourcen. Ressourcen haben einen Typ, welchen wir als Integer repräsentieren. Die Methode **Resource.type()** gibt den Typen einer Ressource zurück. Bauunternehmen und Baustellen gehören Ressourcen. Dabei gehört jede einzelne Ressource immer nur maximal entweder einem einzigen Bauunternehmen oder einer einzigen Baustelle. Der Aufgabentext beschreibt wann und wie sich der Besitz von Ressourcen ändert. Gleichermassen gehört jede Baustelle zu einem einzigen Bauunternehmen. Beim Erstellen der Baustelle wird angegeben, welche Ressourcetypen auf der Baustelle verwendet werden und wie viele Ressourcen des gleichen Typs maximal der Baustelle gehören dürfen.

Die Klasse **CCompany** repräsentiert ein Bauunternehmen (construction company) und hat folgende Methoden:

- **CCompany.resources()** gibt ein Set aller Ressourcen zurück, welche dem Bauunternehmen gehören. Zum Zeitpunkt, wenn ein Bauunternehmen erzeugt wird, gehören dem Bauunternehmen noch keine Ressourcen.
- **CCompany.add(Resource resource)** übergibt dem Bauunternehmen die Ressource **resource**, worauf dem Bauunternehmen die Ressource gehört. Sie dürfen annehmen, dass **resource** vor dem Aufruf keinem anderen Bauunternehmen und keiner Baustelle gehört.
- Die drei Methoden **CCompany.createCSite(Set<Integer> types, int limit)**, **CCompany.createCSite(int type)**, und **CCompany.createCSite(Set<Integer> types, int limit, int flowLimit)** erzeugen eine neue Baustelle, welche dem Bauunternehmen gehört, bis die Baustelle geschlossen wird.
- **CCompany.nextDay()** löst das Übergeben der Ressourcen, welche dem Bauunternehmen gehören, an die Baustellen, welche dem Bauunternehmen gehören, aus. Die Unteraufgaben beschreiben, wie Ressourcen an die Baustellen übergeben werden. Eine übergebene Ressource gehört nicht mehr dem Bauunternehmen (sondern der Baustelle).

Das Interface **CSite** repräsentiert eine Baustelle (construction site) und hat 5 Methoden:

- **CSite.resources()** gibt ein Set aller Ressourcen zurück, welche der Baustelle gehören. Zum Zeitpunkt, wenn eine Baustelle erzeugt wird, gehören der Baustelle noch keine Ressourcen.
- **CSite.canAdd(Resource resource)** gibt zurück, ob der Baustelle die Ressource **resource** übergeben werden darf. Die Unteraufgaben beschreiben, wann das der Fall ist.
- **CSite.add(Resource resource)** übergibt der Baustelle die Ressource **resource**, worauf der Baustelle die Ressource gehört. Die Methode wirft eine **IllegalArgumentException**, falls ein Aufruf von **CSite.canAdd(resource)** **false** zurückgibt. Für die Tests dürfen Sie annehmen, dass **resource**

vor dem Aufruf keinem anderen Bauunternehmen und keiner Baustelle gehört und dass die Methode nicht mehr aufgerufen wird, nachdem durch `CCompany.nextDay()` für die Baustelle die Übergabe von Ressourcen ausgelöst wurde.

- `CSite.use(Resource resource)` verbraucht die Ressource `resource`, welche der Baustelle gehört. Die Methode wirft eine `IllegalArgumentException`, falls das Argument nicht der Baustelle gehört. Nachdem eine Ressource verbraucht wurde, gehört die Ressource nicht mehr der Baustelle (und die Ressource gehört auch nichts anderem mehr).
- `CSite.close()` schliesst eine Baustelle, das heisst, die Baustelle wird vom Bauunternehmen entfernt. Sie dürfen annehmen, dass keine weiteren Methoden auf einer geschlossenen Baustelle aufgerufen werden. Die Ressourcen, welche der Baustelle gehören, ändern sich nicht.

Sie dürfen annehmen, dass, solange die Aufgabenstellung nichts anderes vorgibt, alle Argumente nie `null` sind.

Figure 1a zeigt ein Bauunternehmen Q mit 2 Baustellen X und Y, wobei X vor Y erstellt wurde; das Lager vom Bauunternehmen Q enthält 4 Typen von Ressourcen (T_a , T_b , T_c , T_d). Jedes graue Quadrat stellt eine Ressource dar, wobei die Zahlen angeben in welcher Reihenfolge die Ressourcen an das Bauunternehmen Q übergeben wurden. Die grauen Quadrate bei der Baustelle repräsentieren Ressourcen, welche schon der Baustelle gehören. Auf der Baustelle X werden nur Ressourcen vom Typ T_a , T_c und T_d verwendet (bei Baustelle Y sind es T_a , T_b und T_c), wobei pro Typ maximal zwei Ressourcen der Baustelle X gehören dürfen (bei Baustelle Y sind es pro Typ maximal drei).

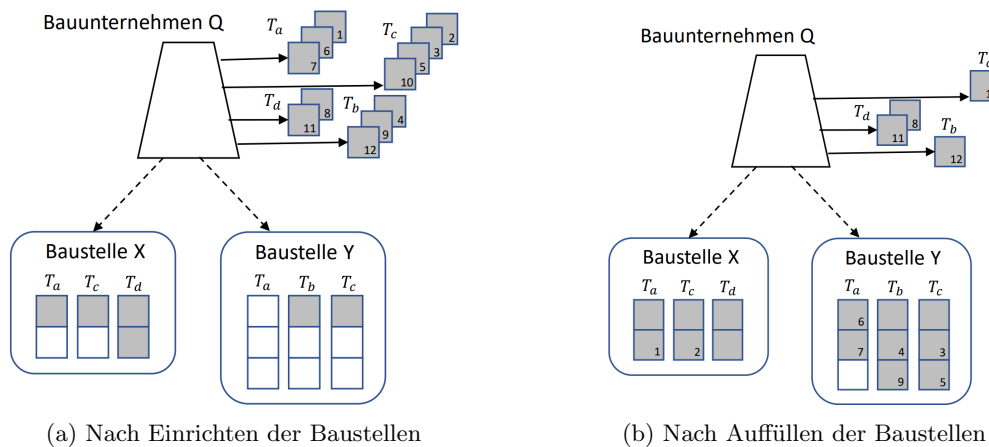


Figure 1: Bauunternehmen (Q) mit 2 Baustellen (X und Y) und den 4 Typen (T_a , T_b , T_c , T_d).

(a) Implementieren Sie die Methoden

- `CCompany.createCSite(Set<Integer> types, int limit)` (Sie dürfen annehmen `limit ≥ 0`). Diese Methode gibt eine Baustelle `site` mit Platz für `limit` Ressourcen jedes Typs zurück. Eine Ressource `resource` darf der Baustelle `site` übergeben werden (das heisst `site.canAdd(resource)` gibt `true` zurück) genau dann, wenn:
 - (i) die Ressource `resource` einen Typ hat, welcher im Set `types` enthalten ist, *und*
 - (ii) falls die Anzahl der Ressourcen, welche der Baustelle gehören und den gleichen Typ wie die Ressource `resource` haben, strikt kleiner als `limit` ist.
- `CSite.canAdd(Resource resource)`,
- `CSite.add(Resource resource)`,

- `CSite.use(Resource resource)`, und
 - `CSite.resources()`.
- (b) Implementieren Sie `CSite.close()`, `CCompany.resources()`, `CCompany.add(Resource resource)`, und `CCompany.nextDay()`. Ein Aufruf von `CCompany.nextDay()` sorgt dafür, dass die Ressourcen, welche dem Bauunternehmen gehören, an die Baustellen, welche dem Bauunternehmen gehören, verteilt werden. Ressourcen werden dabei wie folgt an die Baustellen verteilt: Die Ressourcen werden in der Reihenfolge an die Baustellen übergeben, in welcher die Ressourcen an das Bauunternehmen übergeben wurden. In dieser Reihenfolge wird eine Ressource `resource` an die Baustelle `site` übergeben, falls:
- (i) die Ressource `resource` der Baustelle `site` übergeben werden darf, das heisst der Aufruf `site.canAdd(resource)` gibt `true` zurück, *und*
 - (ii) falls die Ressource `resource` an keine andere Baustelle `otherSite`, welche dem Bauunternehmen gehört und **vor** der Baustelle `site` erzeugt wurde, übergeben werden darf, das heisst der Aufruf `otherSite.canAdd(resource)` gibt `false` zurück.

Beachten Sie, dass es sein kann, dass eine Ressource an keine Baustelle übergeben wird (diese Ressource verbleibt im Lager und kann vielleicht später übergeben werden).

Figure 1b zeigt den Zustand nach Ausführen der Methode `Q.nextDay()`, wobei nach Einrichten der Baustellen (Figure 1a) und dem Aufruf von `nextDay()` keine Ressourcen hinzugefügt oder verbraucht wurden.

Falls Sie Unteraufgabe (a) nicht gelöst haben, so können Sie Teilpunkte bekommen. Eine Teilmenge der Tests wird nur die Methode `CCompany.createCSite(int type)` zum Erzeugen von Baustellen verwenden. Diese Methode gibt eine Baustelle zurück, welche eine Ressource nur annehmen darf, falls der Typ der Ressource gleich `type` ist und der Baustelle nicht schon 2 Ressourcen gehören. Um Teilpunkte zu bekommen, können Sie die Methode `CCompany.createCSite(int type)` mit Hilfe der Klasse `TaskBCSite` implementieren, welche bereits alle Methoden der Unteraufgabe (a) implementiert. Beachten Sie, dass Sie die Klasse `TaskBCSite` verändern dürfen.

- (c) Implementieren Sie die Methode `CCompany.createCSite(Set<Integer> types, int limit, int flowLimit)` (Sie dürfen annehmen $\text{limit} \geq 0$ and $\text{flowLimit} \geq 0$). Die Methode gibt, wie bei Unteraufgabe (a), eine Baustelle `site` zurück, welcher eine Ressource `resource` nur übergeben werden darf (das heisst `site.canAdd(resource)` gibt `true` zurück) genau dann, wenn:
- (i) die Ressource `resource` einen Typ hat, welcher im Set `types` enthalten ist,
 - (ii) und falls die Anzahl der Ressourcen, welche der Baustelle gehören und den gleichen Typ wie die Ressource `resource` haben, strikt kleiner als `limit` ist.

Wir nennen die Baustellen, welche von dieser Methode zurückgegeben werden, “dynamische” Baustellen. Eine dynamische Baustelle `site` hat einen “Überfluss” von einem Typ `type`, falls die Anzahl der Ressourcen, welche der Baustelle `site` gehört und Typ `type` hat, strikt grösser als `flowLimit` ist. Dynamische Baustellen erweitern was bei einem Aufruf von `CCompany.nextDay()` passiert: **Bevor die Ressourcen verteilt werden**, übergibt jede dynamische Baustelle `site`, welche dem Bauunternehmen gehört, dem Bauunternehmen die kleinstmögliche Menge an Ressourcen, sodass die Baustelle `site` keinen Überfluss hat. Im Allgemeinen kann es mehrere Mengen geben, welche dieses Kriterium erfüllen und daher gültige Lösungen sind. Eine Ressource, welche dem Bauunternehmen von einer Baustelle übergeben wird, gehört danach dem Bauunternehmen und nicht mehr der Baustelle. Danach werden die Ressourcen verteilt, wie in Unteraufgabe (b) beschrieben.

Tests finden Sie in der Datei “ConstructionCompanyTest.java”. Die Datei “GradingConstructionCompanyTest.java” enthält die Tests, welche wir an der Prüfung für die Korrektur verwendet haben. Wir empfehlen diese Tests erst zu verwenden, wenn Sie denken, dass Ihre Lösung korrekt ist, damit sie sehen können, wie Sie an einer Prüfung abgeschnitten hätten.