AUFGABENSTELLUNG

ALLGEMEINE INFORMATIONEN

Die Aufgabenstellung ist als Einzelarbeit innerhalb von 2,5 Stunden zu lösen. Wie im realen Programmieralltag ist es erlaubt, das Internet zur Recherche zu verwenden. Bei Fragen zur Problemstellung, wende Dich bitte an einen der KNAPP-Betreuer.

EINLEITUNG

In einem Verteilzentrum werden die einzelnen Produkte in Behältern mit identen Abmessungen gelagert und während des Zusammenpackens von Kundenaufträgen aus diesen entnommen.

Einige der Behälter sind unterteilt, um mehr als ein unterschiedliches Produkt in einem Behälter lagern zu können. Diese Unterteilungen nennt man *Slots.* Ist ein Behälter nicht unterteilt, so besteht er aus einem einzelnen Slot.

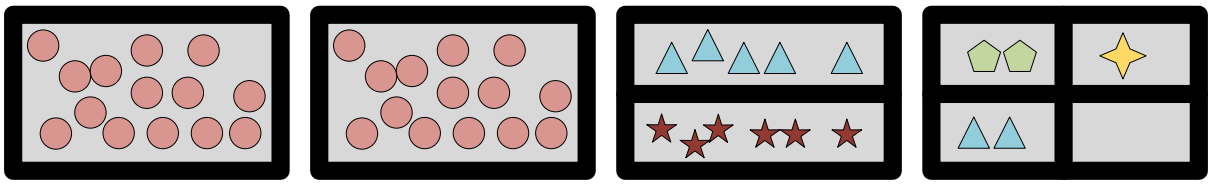

Abbildung 1 - Mögliche Lagerungen (Beispiel)

Da mehrere Kundenaufträge gleichzeitig zusammengepackt werden, kommt es vor, dass ein Produkt an mehreren Orten gleichzeitig gebraucht wird. Wenn dieses Produkt in mehreren Behältern gelagert wird, werden mehrere Behälter auch benutzt. Es wird also nicht zuerst ein Behälter vollständig geleert und dann erst der nächste Behälter verwendet.

Dadurch kommt es zu der Situation, dass für ein Produkt mehr Behälter verwendet werden als für die Lagerung der Produkte tatsächlich notwendig. So passen zum Beispiel 6 Stück in einen Behälter, es gibt aber zwei Behälter für dieses Produkt, einen mit 4 Stück und einen mit 2 Stück.

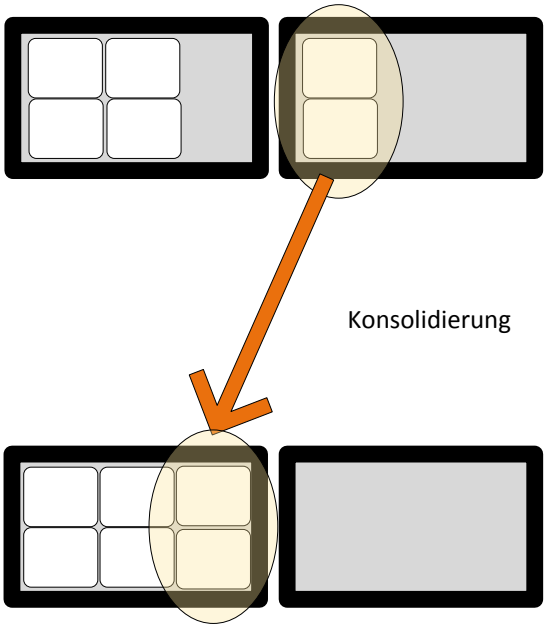

Abbildung 2 - Schema Konsolidierung

Um dies zu berichtigen, wird an einem Arbeitsplatz eine so genannte *Konsolidierung* durchgeführt. Dabei wird dasselbe Produkt aus mehreren Slots auf einen zusammengeführt. Es werden also die 2 Stück in den Slot mit 4 Stück umgepackt, und es entstehen ein voller und ein leerer, wieder verwendbarer Slot.

Damit wird die Anzahl der belegten Slots und Behälter verringert und es stehen im Idealfall mehr Slots und Behälter für neue Produkte zur Verfügung.

AUFGABE

Deine Aufgabe ist es, ein Software-Modul zu entwickeln, das diese Konsolidierung durchführt und somit möglichst viele leere Behälter und Slots entstehen lässt. Dabei

- darf immer nur ein Produkt in einem Slot gelagert werden
- dürfen nur so viel Stück eines Produktes in einem Slot gelagert werden wie Platz finden
- dürfen immer nur maximal 5 Behälter gleichzeitig am Konsolidierungs–Arbeitsplatz stehen
- sollten möglichst viele leere Slots entstehen
- sollen zusätzlich möglichst viele leere Behälter entstehen (alle Slots leer)

ERLÄUTERUNGEN

Produkt

Produkte (*Product*) sind Waren, die gehandelt werden und voneinander unterscheidbar sind. Ein rotes Hemd, Größe 42 ist ein Produkt, es unterscheidet sich von einem roten Hemd Größe 43.

Ein Produkt hat einen Code, der es eindeutig identifiziert und eine maximale Anzahl an Stück die in einen gewissen Typ von Slot passen (**`Product::getMaxQuantity( ContainerType )`**).

Slot

Ein Slot ist eine Unterteilung eines Behälters in dem Produkte aufbewahrt werden können. In einem Slot dürfen sich immer nur Stück desselben Produktes befinden.

Behälter

Ein Behälter (*Container*) dient zur Aufbewahrung und zum Transport von Produkten im Lager. Er ist je nach Typ in eine nicht veränderbare Anzahl von Slots unterteilt. Ein Behälter ohne Unterteilung hat dabei immer einen Slot. Es können Slots mit unterschiedlichen Produkten am selben Behälter sein.

In dieser Aufgabe gibt es drei Unterschiedliche Behältertypen,

1 Slot	ContainerTyp.Full (Behältertyp 1/1)
2 Slots	ContainerTyp.Half (Behältertyp ½)
4 Slots	ContainerTyp.Quarter (Behältertyp ¼)

Ein Behälter hat einen Code, der ihn eindeutig identifiziert. Ein Slot wird über seinen Index im Behälter eindeutig identifiziert (Behälternummer und Index).

ABLAUF

Die Konsolidierung beginnt nachdem viele Kundenaufträge zusammengepackt wurden und die Stück eines Produktes über viele Behälter und Slots verteilt sind. Während der Konsolidierung wird keine andere Arbeit im Verteilzentrum durchgeführt. Außer den durch die Konsolidierung durchgeführten Änderungen kommt es zu keiner Veränderung der Produkte in den Behältern.

Der Bereich in dem die Konsolidierung stattfindet besteht aus dem Behälterlager und einem Arbeitsplatz, an dem die Produkte zwischen den Slots umgepackt werden können. Dieser Arbeitsplatz kann maximal 5 Behälter gleichzeitig aufnehmen. Der Weg zwischen Behälterlager und Arbeitsplatz ist ebenso wie die Position des Behälters am Arbeitsplatz für die Aufgabenstellung nicht relevant.

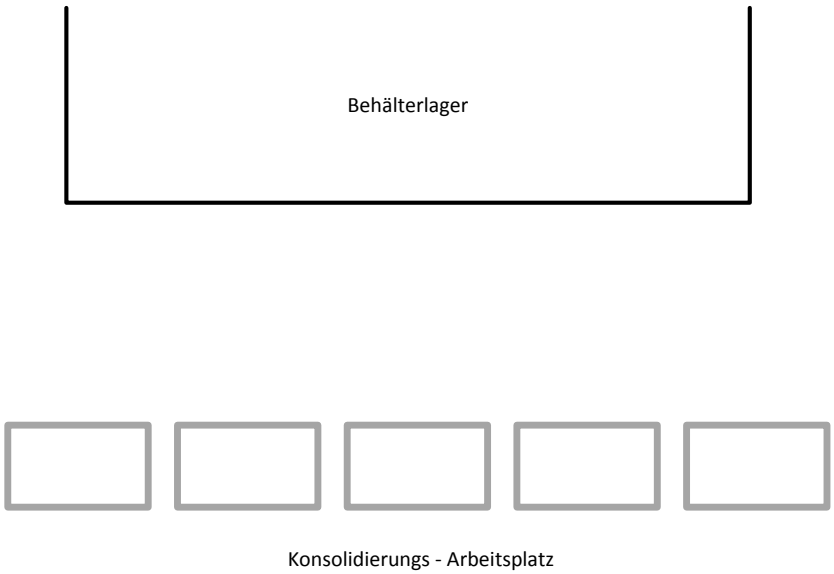

Abbildung 3 – Schema Arbeitsplatz

Zu Beginn der Konsolidierung ist der Arbeitsplatz leer, am Ende dürfen Behälter darin stehen bleiben.

Dir steht eine Liste der Behälter mit deren Slots und deren Inhalt zur Verfügung (**`Container::getSlots`**). Damit kannst du Behälter und Slots finden, deren Produkte Du zusammenlegen kannst.

Um umzupacken musst Behälter aus dem Behälterlager auf den Arbeitsplatz holen. Nach dem Aufruf der Methode (**`Warehouse::moveToWorkStation`**) steht dir der Behälter sofort am Arbeitsplatz zur Verfügung.

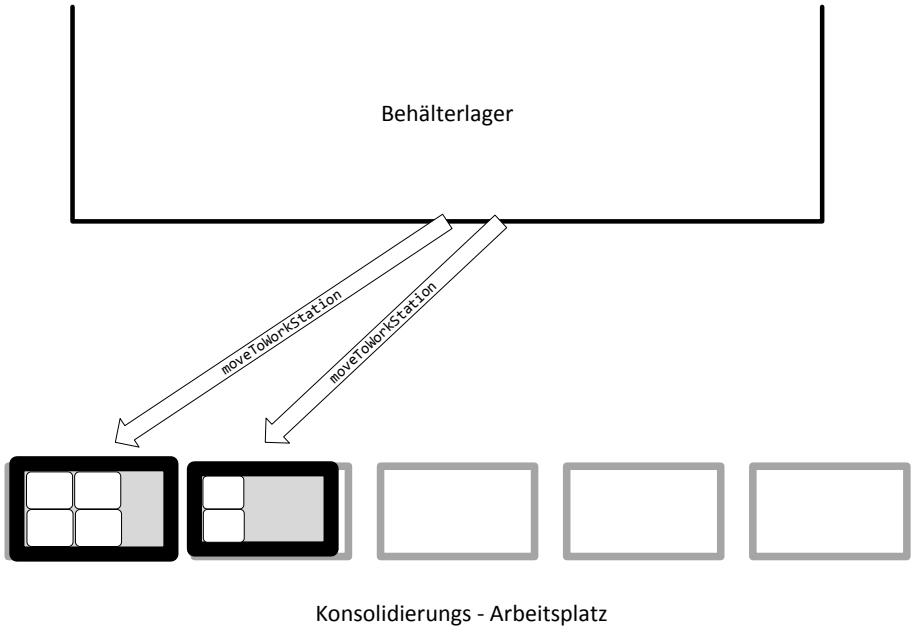

Abbildung 4 - Arbeitsplatz mit geholten Behältern

Wenn die Behälter am Arbeitsplatz stehen, kannst Du Produkte von einem Slot in einen anderen umpacken (**`Warehouse::transferItems`**). Der Typ der beiden Slots muss nicht ident sein, Du kannst also Produkte aus einem ½ Slot in einen ¼ Slot umpacken, sofern dadurch die maximale Stückzahl dieses Produkts pro Slot dieses Typs nicht überschritten wird.

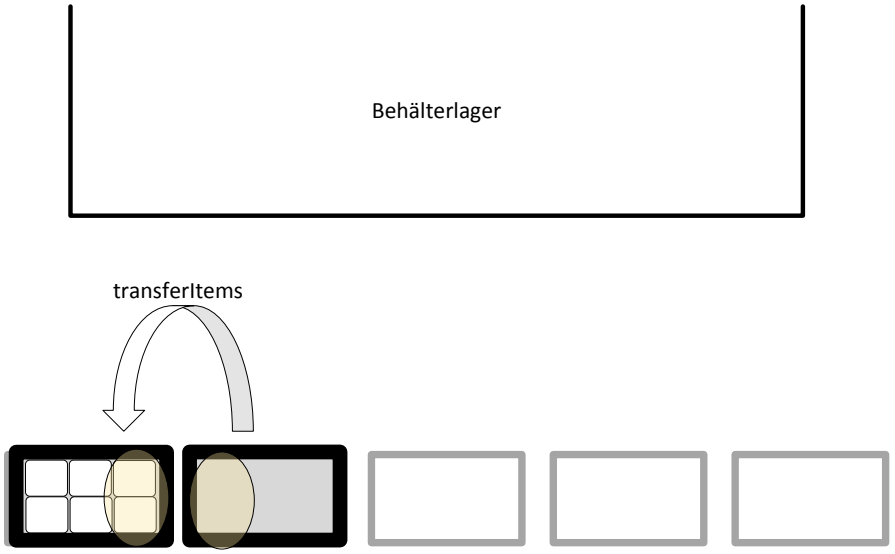

Abbildung 5 - Produkte umpacken

Beim Umlagern musst du auch eine Stückzahl angeben, du könntest also auch den Inhalt eines Slots nur teilweise umpacken, oder den Inhalt eines Slots auf mehrere andere Slots verteilen. Die Behälter können so lange in der Station stehen bleiben wie du sie brauchst.

Wenn du einen Behälter nicht mehr brauchst, kannst du diesen wieder in das Behälterlager stellen (`Warehouse::moveToStorage`). Damit wird ein Platz für einen anderen Behälter am Arbeitsplatz frei.

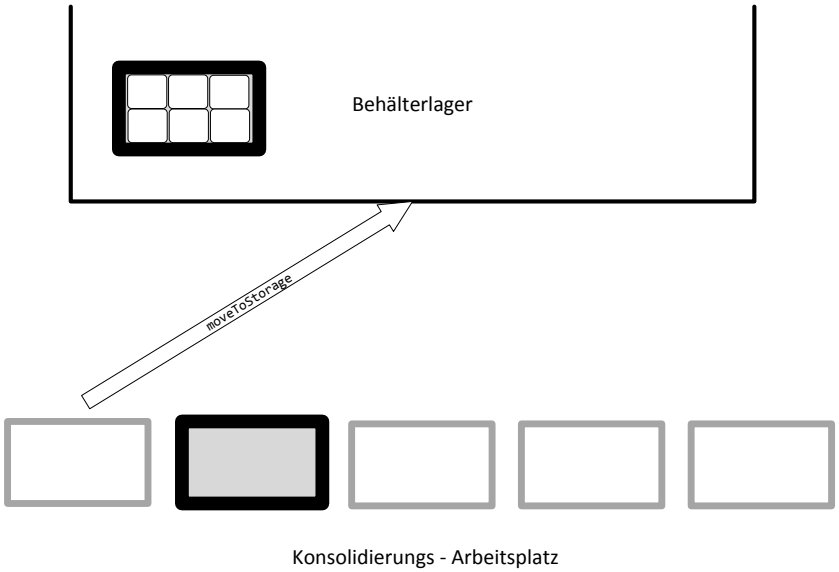

Abbildung 6 - Arbeitsplatz nach Behälterrücklagerung

Beachte, dass die Anzahl an Behälteranforderungen die dir zur Verfügung stehen mit 5000 begrenzt ist (**`Warehouse::CONTAINER_MAX_GET`**).

BEWERTUNGSSCHEMA

- Es werden nur auf den Bewertungsserver hochgeladene Ergebnisse gewertet.
- Die Resultate werden am Server mit den dort hinterlegten Algorithmen errechnet.
    - Die maximale Anzahl von 5.000 geholten Behältern darf nicht überschritten werden.
- Pro leerem Slot gibt es je nach Typ Punkte

1 Slot	ContainerTyp.Full	4 Punkte
1 Slot	ContainerTyp.Half	2 Punkte
1 Slot	ContainerTyp.Quarter	1 Punkt

- Für jeden komplett leeren Behälter gibt es 2 Bonuspunkte.
- Lösungen die früher abgegeben werden sind besser.

Bei mehreren Abgaben (Uploads) zählt immer die beste Abgabe zu dem Zeitpunkt an dem diese getätigt wurde. Wenn Du nach dem Upload einer Lösung weiter arbeitest und durch Deine Änderungen das Ergebnis schlechter wird, bleibt die zweite Abgabe unberücksichtigt.

ABGABEMODUS

Zur Beurteilung des Ergebnisses wird das Ergebnis-CSV-File jedes Teilnehmers von KNAPP interpretiert. Dazu muss die vom KNAPP Code automatisch erzeugte Datei `upload2017.zip` über die Abgabeseite hochgeladen werden.

Du kannst alle Code Teile modifizieren, die Ergebnisdatei (`warehouse-operations.csv`) wird von uns interpretiert und darf daher im Format nicht verändert werden.

UPLOAD WEBSITE

Unmittelbar nach dem Upload erhältst du ein detailliertes Feedback zu deiner Abgabe.

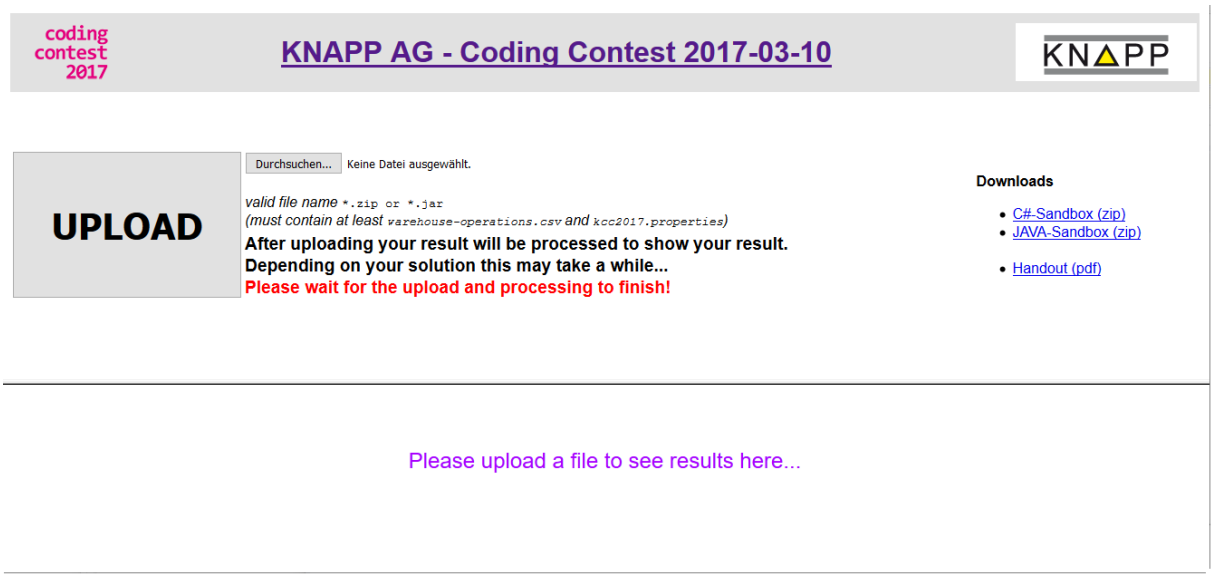

Abbildung 7: Upload-Server (Beispiel)

Im Feedback siehst Du die Probleme, die noch in deiner Abgabe vorhanden sind. Erst wenn alle Muss-Kriterien erfüllt sind, siehst Du auch den Score den deine Lösung erzielt. Je höher dieser Wert ist, desto besser ist das Ergebnis.

TIPPS

- Versuche mit Deiner Software das Ergebnis zuerst mit einfachen Strategien zu verbessern und arbeite danach an weiteren Optimierungen.
- Schau Dir den von uns vorgegeben Code an und prüfe, ob Du Teile wiederverwenden oder erweitern kannst.
- Du kannst alle Teile des Source-Codes verändern. Die Abgabedatei muss jedoch dem vorgegebenen Format entsprechen.
- Lade öfters hoch - es wird nur die beste Abgabe bewertet.

CODE-DETAILS

Folgende Daten stehen Dir von Beginn an zur Verfügung:

1. alle Produkte (*products*)
2. alle Behälter und deren Inhalte (containers)

Deine Startpunkt ist die Methode **`optimize()`** in der Klasse **`solution`**::**`OptimizeStorage.`**

KLASSENDIAGRAMM

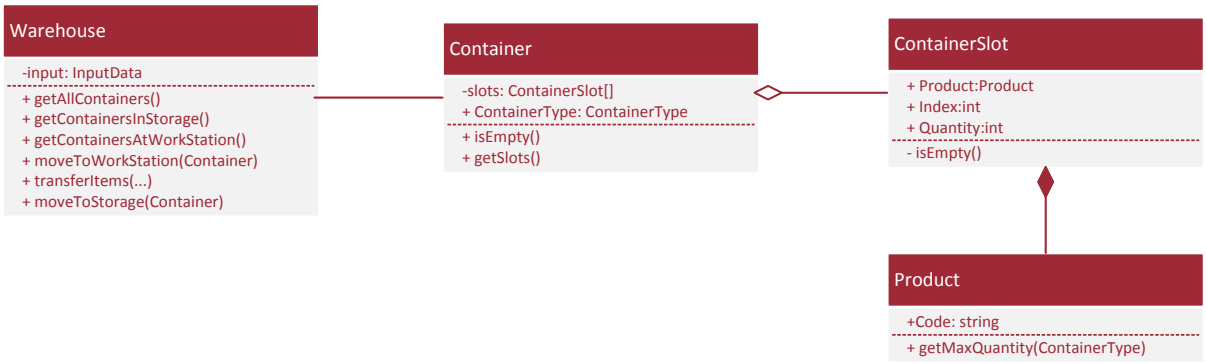

Abbildung 8: Klassendiagramm

..:: PLATZ FÜR NOTIZEN ::..