Karlsruher Institut für Technologie

Programmieren – Wintersemester 2018/19

Prof. Dr. Ralf H. Reussner
Software-Entwurf und -Qualität (SDQ)
`https://sdqweb.ipd.kit.edu/wiki/Programmieren`
`programmieren-vorlesung@ipd.kit.edu`

Abschlussaufgabe 2

Ausgabe: 22.02.2019 – 13:00 Uhr
Abgabe: 23.03.2019 – 06:00 Uhr

Bearbeitungshinweise

- Achten Sie darauf, nicht zu lange Zeilen, Methoden, Klassen und Schnittstellen zu erstellen[1].
- Programmcode muss in englischer Sprache verfasst sein.
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich. Die Kommentare sollen einheitlich in englischer oder deutscher Sprache verfasst werden.
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute.
- Verwenden Sie keine Klassen der Java-Bibliotheken ausgenommen Klassen der Pakete java.lang, java.util und java.util.regex, es sei denn, die Aufgabenstellung erlaubt ausdrücklich weitere Pakete[1].
- Achten Sie auf fehlerfrei kompilierenden Programmcode[1].
- Halten Sie alle Whitespace-Regeln ein[1].
- Halten Sie die Regeln zu Variablen-, Methoden- und Paketbenennung ein und wählen Sie aussagekräftige Namen[1].
- Halten Sie die Regeln zur Javadoc-Dokumentation ein[1].
- Nutzen Sie nicht das default-Package[1].
- Halten Sie auch alle anderen Checkstyle-Regeln ein[1].
- `System.exit()` und `Runtime.exit()` dürfen nicht verwendet werden[1].
- Halten Sie die Hinweise zur Modellierung im Ilias-Wiki ein.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Freitag, den 08.03.2019 um 13:00 Uhr**, freigeschaltet.
Laden Sie die `Terminal`-Klasse nicht mit hoch.

- Geben Sie Ihre Klassen zu Abschlussaufgabe 2 als `*.java`-Dateien ab.

Plagiarismus
Das Einreichen fremder Lösungen, seien es auch teilweise Lösungen von Dritten, aus Büchern, dem Internet oder anderen Quellen wie beispielsweise der Lehrveranstaltung oder dem Übungsbetrieb selbst, ist ein Täuschungsversuch und führt zur Bewertung nicht bestanden. Dies beinhaltet unter anderem auch die Lösungsvorschläge des Übungsbetriebes. Es werden nur rein selbstständig erarbeitete Lösungen akzeptiert. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

[1] Der Praktomat wird die Abgabe zurückweisen, falls diese Regel verletzt ist.

Terminal-Klasse
Laden Sie für diese Aufgabe die Terminal-Klasse herunter und platzieren Sie diese unbedingt im Paket `edu.kit.informatik`. Die Methode `Terminal.readLine()` liest eine Benutzereingabe von der Konsole und ersetzt `System.in`. Die Methode `Terminal.printLine()` schreibt eine Ausgabe auf die Konsole und ersetzt `System.out`. Verwenden Sie für jegliche Konsoleneingabe oder Konsolenausgabe die Terminal-Klasse. Verwenden Sie in keinem Fall `System.in` oder `System.out`. Fehlermeldungen werden ausschließlich über die Terminal-Klasse ausgegeben und müssen aus technischen Gründen unbedingt mit `Error,` beginnen. Dies kann unter anderem bequem über die `Terminal.printError`-Methode erfolgen. **Modifizieren Sie niemals die Terminal-Klasse und laden Sie diese auch niemals zusammen mit Ihrer Abgabe hoch.**

Fehlerbehandlung
Ihre Programme sollen auf ungültige Benutzereingaben mit einer aussagekräftigen Fehlermeldung reagieren. Aus technischen Gründen muss eine Fehlermeldung unbedingt mit `Error,` beginnen. Eine Fehlermeldung führt nicht dazu, dass das Programm beendet wird; es sei denn, die nachfolgende Aufgabenstellung verlangt dies ausdrücklich. Achten Sie insbesondere auch darauf, dass unbehandelte `RuntimeExceptions`, bzw. Subklassen davon – sogenannte *Unchecked Exceptions* – nicht zum Abbruch Ihres Programms führen sollen.

Objektorientierte Modellierung
Achten Sie darauf, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung als auch Funktionalität bewertet werden.

Öffentliche Tests
Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe nötig ist. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

Materialbedarfsermittlung (20 Punkte)

In dieser Aufgabe soll ein spezielles System zur Materialbedarfsermittlung in Unternehmen implementiert werden. Diese Materialbedarfsermittlung befasst sich mit der Festlegung und Ermittlung der für die Produktion benötigten Bedarfs an *Baugruppen* und *Einzelteilen*. Hierbei wird dieser Bedarf auf der Basis von *Materialstücklisten* abgeleitet und als *Mengenstückliste* angegeben.

Ein Einzelteil ist ein nicht weiter zerlegbarer Gegenstand. Im Gegensatz hierzu besteht eine Baugruppe aus mindestens einem Einzelteil und/oder aus mindestens einer Baugruppen. Folglich kann es auch Baugruppen geben, welche selbst aus nur Baugruppen oder aus nur Einzelteilen besteht. Auch kann eine Baugruppe wieder Teil verschiedener Baugruppen sein. Es kann aber auch Baugruppen geben, welche keine Verbindungen zu andern Baugruppen haben. Baugruppen müssen aber mindestens aus einem Einzelteile oder aus einer Baugruppe bestehen.

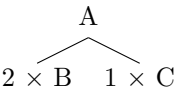
In dem zu implementierenden System zur Materialbedarfsermittlung müssen Einzelteil immer in einer Relation zu einer Baugruppe stehen. Einzelteile die nicht Teil einer Baugruppe sind, werden automatisch aus dem System entfernt.

Materialstücklisten

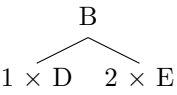
Die Beziehung zwischen Einzelteilen und Baugruppen wird in Materialstücklisten beschrieben. Eine Materialstückliste ist eine strukturierte Anordnung von Baugruppen und/oder Einzelteilen, welche zur Produktion einer anderen übergeordneten Baugruppe benötigt werden. Eine Materialstückliste ordnet so Baugruppen und Ein-

zelteilen nach deren strukturellen Eigenschaften und definiert somit, aus welchen und wie vielen Baugruppen und/oder Einzelteilen eine bestimmte Baugruppen besteht. Hierbei werden die Mengenangaben für eine Baugruppe oder ein Einzelteil, welches direkt in einer Materialstückliste enthalten ist auf die natürlichen Zahlen aus dem abgeschlossenen Intervall von `1` bis `1000` festgelegt: `[1, 1000]`

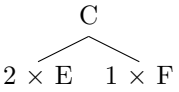
In dem zu implementierenden System wird ein Gegenstand als eine Baugruppe definiert, sobald eine Materialstückliste für diesen definiert wurde. Solange keine Materialstückliste für einen Gegenstand definiert ist, ist dieser als ein Einzelteil definiert. Auch werden alle Baugruppen und Einzelteile eindeutig durch ihren Namen identifiziert. Folglich kann immer nur eine Materialstückliste für eine Baugruppe im System existent sein. Hierbei werden diese eindeutigen Namen auf Zeichenketten festgelegt, welche aus einer Folge von mindestens einem Klein- oder Großbuchstaben von `A` bis `Z` bestehen: `[a-zA-Z]+`



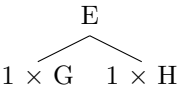
Materialstückliste 1: Strukturelle Eigenschaft der Baugruppe „A“



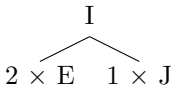
Materialstückliste 2: Strukturelle Eigenschaft der Baugruppe „B“



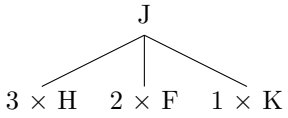
Materialstückliste 3: Strukturelle Eigenschaft der Baugruppe „C“



Materialstückliste 4: Strukturelle Eigenschaft der Baugruppe „E“



Materialstückliste 5: Strukturelle Eigenschaft der Baugruppe „I“



Materialstückliste 6: Strukturelle Eigenschaft der Baugruppe „J“

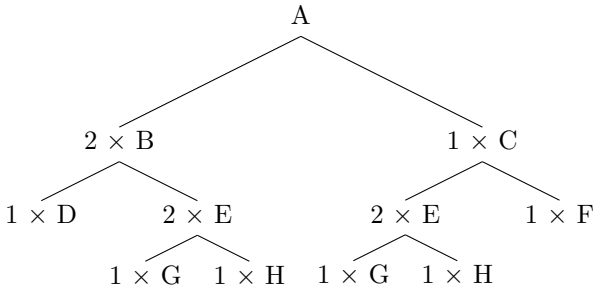
Die Materialstücklisten 1 bis 6 zeigen beispielhaft vier verschiedene Baugruppen: *A*, *B*, *C*, *E*, *I* und *J*. In dieser beispielhaften visuellen Darstellung dieser Materialstücklisten definiert jeweils die oberste Zeichenkette den Namen der Baugruppe. Während in diesen Materialstücklisten jeweils die unteren Zeichenketten definieren, aus welchen und wie vielen Baugruppen und/oder Einzelteilen sich die Baugruppe (oberste Zeichenkette) zusammensetzt. So setzt sich die Baugruppe *A* aus zwei *B*s und einem *C* zusammen. Während sich in diesem Beispiel die Baugruppe *J* wiederum aus drei *H*s, zwei *F*s und einem *K* zusammensetzt.

Produktstruktur

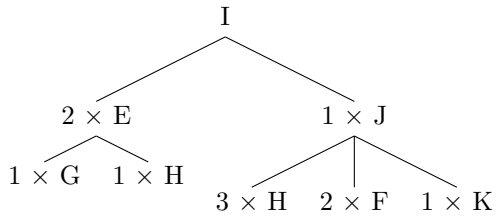
Eine vollständige Produktstruktur setzt sich erst aus hierarchisch angeordneten Materialstücklisten zusammen. Zur Ermittlung des für die Produktion benötigten Bedarfs aller Einzelteile für eine Baugruppe, müssen die im System existierenden Materialstücklisten im ganzen Zusammenhang betrachtet werden.

Die Produktstruktur 1 zeigt beispielhaft für die Baugruppe *A*, wie sich diese aus weiteren Baugruppen vollständig zusammensetzt. Diese Baugruppe *A* besteht direkt aus den Baugruppen *B* und *C* und wiederum indirekt aus der Baugruppe *E*. Hierbei wird die Baugruppe *E* von zwei verschieden Baugruppen (*B* und *C*) verwendet. Zugleich besteht die Baugruppe *A* indirekt aus den Einzelteilen *D*, *F*, *G* und *H*.

Die Produktstruktur 2 zeigt beispielhaft für die Baugruppe *I*, wie sich diese aus weiteren Baugruppen vollständig zusammensetzt. Diese Baugruppe *I* besteht direkt aus den Baugruppen *E* und *J*, sowie aus den Einzelteilen *F*, *G*, *H* und *K*. Hierbei wird das Einzelteil *H* von zwei verschieden Baugruppen (*E* und *J*) verwendet.



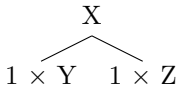
Produktstruktur 1: Vollständige hierarchische Produktstruktur der Baugruppe „A“



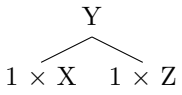
Produktstruktur 2: Vollständige hierarchische Produktstruktur der Baugruppe „I“

Zyklenfreiheit

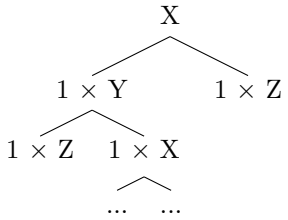
Damit immer eine solche vollständige Produktstruktur ermittelt werden kann, muss von dem zu implementierenden System sichergestellt werden, dass keine Materialstückliste eingegebenen wird, welche einen Zyklus in der Produktstruktur verursacht. Hierbei besteht ein Zyklus dann, wenn eine Baugruppe eine Baugruppe beinhaltet, welche direkt oder indirekt wieder diese Baugruppe beinhaltet. Das heißt, in der Produktstruktur existiert ein Weg von einer Baugruppe wieder zu dieser. Folglich darf eine Baugruppe auch nicht sich direkt oder indirekt selbst wieder zum Inhalt haben.



Materialstückliste 7: Baugruppe „X“



Materialstückliste 8: In Verbindung mit Baugruppe „X“ unzulässige Baugruppe „Y“



Produktstruktur 3: Unzulässige Zyklische Produktstruktur der Baugruppe „X“

Beispielsweise zeigt die Materialstückliste 8 in Verbindung mit der Materialstückliste 7 eine nicht erlaubte Materialstückliste, da diese einen Zyklus in der Produktstruktur verursacht. Da die Baugruppe „X“ die Baugruppe „Y“ beinhaltet und diese Baugruppe „Y“ wiederum die Baugruppe „X“ beinhaltet, verletzt diese Konstellation der beiden Baugruppe die geforderte Zyklenfreiheit. Die Produktstruktur 3 zeigt beispielhaft diese im System nicht erlaubte zyklische Produktstruktur der Baugruppe „X“. In dem zu implementierenden System muss bei jeder Definition einer neuen Materialstückliste geprüft werden, ob durch diese die geforderte Zyklenfreiheit verletzt würde.

Mengenstücklisten

Eine der elementarsten Fragen bei der Materialbedarfsermittlung ist, welche Einzelteile in welcher Menge dazu benötigt werden um eine Baugruppen zu produzieren. Die Antwort auf diese Frage liefert die Mengenstückliste. Eine Mengenstückliste leitet sich aus einer Produktstruktur ab, indem man die Produktstruktur von oben nach unten traversiert. Auch wenn die Mengenangaben für eine Baugruppe oder ein Einzelteil in einer Materialstückliste auf maximal 1000 begrenzt ist, sind die Mengenangaben für eine Baugruppe oder ein Einzelteil in einer Mengenstückliste nicht begrenzt. In einer Mengenstückliste werden im Sinn einer unstrukturierten Aufzählung alle Baugruppen und Einzelteile mit ihrer Menge genannt, welche in einer Baugruppe direkt und indirekt verbaut sind.

Baugruppe: A			
Baugruppe		Einzelteil	
Name	Menge	Name	Menge
E	6	G	6
B	2	H	6
C	1	D	2
		F	1

Mengenstückliste 1: Mengenstücklisten der Baugruppe „A“

Baugruppe: I			
Baugruppe		Einzelteil	
Name	Menge	Name	Menge
E	2	H	5
J	1	F	2
		G	2
		K	1

Mengenstückliste 2: Mengenstücklisten der Baugruppe „I“

Die Mengenstücklisten 1 und 2 zeigen beispielhaft für die Baugruppen A und I, wie sich diese vollständig aus ihren beiden Posten (Baugruppen und Einzelteile) zusammensetzen. Hierbei werden für jeden Posten dieser beiden Baugruppen die Mengen jeweils auf Grundlage der Produktstruktur ermittelt. Diese beiden Mengenstücklisten zeigen so jeweils für die Baugruppen A und I, aus welchen und wie vielen Baugruppen und Einzelteile sich diese jeweils zusammensetzt.

Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Programm über die Konsole mittels `Terminal.readLine()` verschiedene Arten von Befehlen entgegen, die im Folgenden näher spezifiziert werden. Nach Abarbeitung eines Befehls wartet Ihr Programm auf weitere Befehle, bis das Programm irgendwann durch den `quit`-Befehl beendet wird. Alle Befehle werden auf dem aktuellen Zustand des Systems zur Materialbedarfsermittlung ausgeführt.

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die zuvor definierten Grundlagen und Bedingungen nicht verletzt werden und geben Sie in diesen Fällen immer eine aussagekräftige Fehlermeldung aus. So muss unter anderem beim Hinzufügen einer neuen Baugruppe oder bei dem Verändern einer bereits bestehenden Baugruppe stets darauf geachtet werden, dass keine Anforderung, wie beispielsweise die Vorgaben für den Namen und die Menge oder die geforderte Zyklenfreiheit verletzt werden. Entspricht eine Eingabe nicht dem vorgegebenen Format, dann ist immer eine Fehlermeldung auszugeben. Danach soll das Programm auf die nächste Eingabe warten. Bei Fehlermeldungen dürfen Sie den Text frei wählen, er sollte jedoch sinnvoll sein. Jede Fehlermeldung muss aber mit `Error, ` beginnen und darf keine Zeilenumbrüche enthalten.

Da wir automatische Tests Ihrer interaktiven Benutzerschnittstelle durchführen, müssen die Ausgaben exakt den Vorgaben entsprechen. Insbesondere sollen sowohl Klein- und Großbuchstaben als auch die Leerzeichen und Zeilenumbrüche genau übereinstimmen. Geben Sie auch keine zusätzlichen Informationen aus. Beginnen Sie frühzeitig mit dem Einreichen, um Ihre Lösung dahingehend zu testen, und verwenden Sie das Forum, um eventuelle Unklarheiten zu klären.

Beachten Sie, dass bei der Beschreibung der Eingabe- und Ausgabeformate die Wörter zwischen spitzen Klammern (`<` und `>`) für Platzhalter stehen, welche bei der konkreten Ein- und Ausgabe durch Werte ersetzt werden. Diese eigentlichen Werte enthalten bei der Ein- und Ausgabe keine Klammern. Vergleichen Sie hierzu auch die jeweiligen Beispielabläufe.

Beachten Sie auch, dass bei den folgenden Beispielabläufen die Eingabezeilen mit dem `>` -Zeichen gefolgt von einem Leerzeichen eingeleitet werden. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabezeilen.

Die in diesem Abschnitt folgenden einzelnen Beispielabläufen lassen sich als ein großer zusammenhängender Beispielablauf betrachten.

Der `addAssembly`-Befehl

Legt eine neue Materialstückliste für die Baugruppe mit dem angegebenen Namen (`<nameAssembly>`) in dem System an. An dieser Stelle wird auch mit eingegeben, aus welchen (`<name>`) und wie vielen (`<amount>`)

Baugruppen und/oder Einzelteilen diese Baugruppe besteht. Dabei wird der Name und die Menge durch einen Doppelpunkt (`:`) getrennt Hierbei können ein oder mehrere Baugruppen/Einzelteile nach dem spezifizierten Eingabeformat angegeben werden, welche jeweils durch ein Semikolon (`;`) getrennt werden. Dabei darf aber kein Name in dieser Materialstückliste doppelt vorkommen.

Eingabeformat

`addAssembly <nameAssembly>=<amount`$_1$`>:<name`$_1$`>;<amount`$_2$`>:<name`$_2$`>;...;<amount`$_n$`>:<name`$_n$`>`

Ausgabeformat

Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.B., wenn bereits eine Baugruppe mit dem angegebenen Namen existiert, wenn die Zyklenfreiheit verletzt wird oder bei fehlerhaften Namen, Mengenangaben oder Eingabeformat usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `addAssembly`-Befehl

```
> addAssembly A=2:B;1:C
OK
> addAssembly B=1:D;2:E
OK
> addAssembly C=2:E;1:F
OK
> addAssembly E=1:G;1:H
OK
> addAssembly I=2:E;1:J
OK
> addAssembly J=3:H;2:F;1:K
OK
> addAssembly X=1:Y;1:Z
OK
> addAssembly Y=1:X;1:Z
Error, the specified BOM Y would create a cycle in the product structure: X-Y-X
> addAssembly Y=1:Y;1:Z
Error, the specified BOM Y would create a cycle in the product structure: Y-Y
> addAssembly Y=1:Z;1:Z
Error, the names of the parts in the specified BOM Y appear twice: Z
> addAssembly Y=2:Z
OK
```

Der `removeAssembly`-Befehl

Entfernt eine bestehende Materialstückliste für die Baugruppe mit dem angegebenen Namen (`<nameAssembly>`) aus dem System. Wenn eine Baugruppe entfernt wird und diese zuvor Teil einer anderen Baugruppe war, wird diese in dieser bestehenden Baugruppe anschließend als Einzelteil betrachtet. Dementsprechend entfernt dieser Befehl nur die angegebene Baugruppe und verändert keine weiteren Baugruppen.

Eingabeformat

`removeAssembly <nameAssembly>`

Ausgabeformat

Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.B., wenn keine Baugruppe mit dem angegebenen Namen existiert, bei fehlerhaften Eingabeformat usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `removeAssembly`-Befehl

```
> removeAssembly Y
OK
> removeAssembly Y
Error, no BOM exists in the system for the specified name: Y
```

Der `printAssembly`-Befehl

Gibt für eine im System bestehende Baugruppe mit dem angegebenen Namen (`<nameAssembly>`) ihre Materialstückliste in einer Zeile aus. Dabei werden nur die direkt enthaltenen Baugruppen und/oder Einzelteile für diese Materialstückliste ausgegeben. Diese Baugruppen/Einzelteile werden jeweils durch ein Semikolon (`;`) getrennt. Der Name (`<name>`) einer Baugruppen oder eines Einzelteils und die zugehörige Menge (`<amount>`) werden jeweils durch einen Doppelpunkt (`:`) getrennt. Hierbei werden die Baugruppen/Einzelteile nach ihrem Namen alphabetisch aufsteigend (kleinster Wert zuerst) sortiert, wobei diese lexikografische Sortierung auf dem Unicode-Wert jedes Zeichens in den Zeichenketten basiert und folglich dabei die Groß- und Kleinschreibung beachtet.

Wenn es sich bei dem angegebenen Namen um ein Einzelteil handelt, wird lediglich `COMPONENT` in einer Zeile ausgegeben.

Eingabeformat

`printAssembly <nameAssembly>`

Ausgabeformat

$\texttt{<name}_1\texttt{>:<amount}_1\texttt{>;<name}_2\texttt{>:<amount}_2\texttt{>;...;<name}_n\texttt{>:<amount}_n\texttt{>}$ Im Fehlerfall (z.B., wenn keine Baugruppe/Einzelteil mit dem angegebenen Namen existiert, bei fehlerhaften Eingabeformat usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `printAssembly`-Befehl

```
> printAssembly A
B:2;C:1
> printAssembly B
D:1;E:2
> printAssembly E
G:1;H:1
> printAssembly J
F:2;H:3;K:1
> printAssembly K
COMPONENT
> printAssembly Y
COMPONENTError, no BOM or component exists in the system for the specified name: Y
```

Der `getAssemblies`-Befehl

Gibt für eine im System bestehende Baugruppe mit dem angegebenen Namen (`<nameAssembly>`) die Baugruppen aus ihrer Mengenstückliste in einer Zeile aus. Hierbei werden alle Baugruppen mit ihrer Anzahl ausgegeben, welcher in der angegeben Baugruppe direkt und indirekt verbaut werden. Diese Baugruppen werden jeweils durch ein Semikolon (`;`) getrennt und der Name (`<name>`) und die zugehörige Menge (`<amount>`) einer Baugruppen werden jeweils durch einen Doppelpunkt (`:`) getrennt. Dabei werden diese Baugruppen jeweils absteigend (größter Wert zuerst) nach ihrer Menge sortiert. Bei gleicher Menge, wird anschließend nach Namen der Baugruppen alphabetisch aufsteigend (kleinster Wert zuerst) sortiert, wobei diese lexikografische Sortierung auf dem Unicode-Wert jedes Zeichens in den Zeichenketten basiert und folglich dabei die Groß- und Kleinschreibung beachtet.

Wenn die angegebene Baugruppe nur aus Einzelteilen besteht, wird lediglich `EMPTY` in einer Zeile ausgegeben.

Eingabeformat

`getAssemblies <nameAssembly>`

Ausgabeformat

$\texttt{<name}_1\texttt{>:<amount}_1\texttt{>;<name}_2\texttt{>:<amount}_2\texttt{>;...;<name}_n\texttt{>:<amount}_n\texttt{>}$ Im Fehlerfall (z.B., wenn keine Baugruppe mit dem angegebenen Namen existiert, bei fehlerhaften Eingabeformat usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `getAssemblies`-Befehl

```
> getAssemblies A
E:6;B:2;C:1
> getAssemblies I
E:2;J:1
> getAssemblies E
EMPTY
```

Der `getComponents`-Befehl

Gibt für eine im System bestehende Baugruppe mit dem angegebenen Namen (`<nameAssembly>`) die Einzelteile aus ihrer Mengenstückliste in einer Zeile aus. Hierbei werden alle Einzelteile mit ihrer Anzahl ausgegeben, welcher in der angegeben Baugruppe direkt und indirekt verbaut werden. Diese Einzelteile werden jeweils durch ein Semikolon (`;`) getrennt und der Name (`<name>`) und die zugehörige Menge (`<amount>`) eines Einzelteils wird jeweils durch einen Doppelpunkt (`:`) getrennt. Dabei werden diese Einzelteile jeweils absteigend (größter Wert zuerst) nach ihrer Menge sortiert. Bei gleicher Menge, wird anschließend nach Namen des Einzelteils alphabetisch aufsteigend (kleinster Wert zuerst) sortiert, wobei diese lexikografische Sortierung auf dem Unicode-Wert jedes Zeichens in den Zeichenketten basiert und folglich dabei die Groß- und Kleinschreibung beachtet.

Eingabeformat

`getComponents <nameAssembly>`

Ausgabeformat

$\texttt{<name}_1\texttt{>:<amount}_1\texttt{>;<name}_2\texttt{>:<amount}_2\texttt{>;...;<name}_n\texttt{>:<amount}_n\texttt{>}$ Im Fehlerfall (z.B., wenn keine Baugruppe mit dem angegebenen Namen existiert, bei fehlerhaften Eingabeformat usw.) wird eine aussagekräftige Fehler-

meldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `getComponents`-Befehl

```
> getComponents A
G:6;H:6;D:2;F:1
> getComponents I
H:5;F:2;G:2;K:1
> getComponents E
G:1;H:1
```

Der `addPart`-Befehl

Fügt der bestehenden Materialstücklisten für die Baugruppe mit dem angegebenen Namen (`<nameAssembly>`), die angegebene Menge (`<amount>`) einer Baugruppe oder eines Einzelteils mit dem angegebenen Namen (`<name>`). Wenn die Baugruppe (`<nameAssembly>`) bereits die Baugruppe oder das Einzelteil (`<name>`) enthält, wird die angegebene Menge (`<amount>`) auf die bereits zuvor in der Materialstückliste definierte Menge aufaddiert. Falls die Baugruppe oder das Einzelteil (`<name>`) noch nicht enthalten ist, wird dieses mit der angegebenen Menge der bestehenden Materialstückliste für die Baugruppe (`<nameAssembly>`) hinzugefügt. Dabei muss darauf geachtet werden, dass die zuvor definierten Grundlagen und Bedingungen nicht verletzt werden.

Eingabeformat

`addPart <nameAssembly>+<amount>:<name>`

Ausgabeformat

Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.B., wenn keine Baugruppe mit dem angegebenen Namen existiert, wenn die Zyklenfreiheit verletzt wird oder bei fehlerhaften Namen, Mengenangaben oder Eingabeformat usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `addPart`-Befehl

```
> addPart X+1:A
OK
> addPart X+2:A
OK
> addPart X+1:X
Error, adding the part X to the BOM X would create a cycle in the structure: X-X
> addPart Y+1:A
Error, no BOM exists in the system for the specified name: Y
> addPart E+1:A
Error, adding the part A to the BOM E would create a cycle in the structure: A-B-E-A
```

Der `removePart`-Befehl

Entfernt aus der bestehenden Materialstücklisten für die Baugruppe mit dem angegebenen Namen (`<nameAssembly>`), die angegebene Menge (`<amount>`) einer Baugruppe oder eines Einzelteils mit dem angegebenen Namen (`<name>`). Wenn die Baugruppe (`<nameAssembly>`) die Baugruppe oder das Einzelteil (`<name>`) in mindestens der angegebenen Menge (`<amount>`) enthält, wird diese angegebene Menge von der

bereits zuvor in der Materialstückliste definierte Menge subtrahiert. Falls die angegebene Menge (`<amount>`) und die zuvor in der Materialstückliste definierte Menge gleich sind, wird die Baugruppe oder das Einzelteil (`<name>`) vollständig aus der der bestehende Materialstücklisten entfernt. Dabei muss darauf geachtet werden, dass die zuvor definierten Grundlagen und Bedingungen nicht verletzt werden.

Falls eine Baugruppe, nachdem erfolgreichem Entfernen einer Baugruppe oder eines Einzelteils aus dieser Baugruppe, keine weiteren Baugruppen/Einzelteile mehr enthält, wird die Materialstücklisten für diese Baugruppe automatisch aus dem System entfernt. Wenn diese zuvor Teil einer anderen Baugruppe war, wird diesen in dieser bestehenden Baugruppe anschließend als Einzelteil betrachtet. Dementsprechend entfernt dieser Befehl nur die angegebene Menge und gegebenenfalls die Baugruppe aber verändert keine weiteren Baugruppen.

Eingabeformat

```
removePart <nameAssembly>-<amount>:<name>
```

Ausgabeformat

Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall (z.B., wenn keine Baugruppe mit dem angegebenen Namen existiert, wenn die angegebene Menge nicht mindestens enthalten ist oder bei fehlerhaften Namen, Mengenangaben oder Eingabeformat usw.) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `removePart`-Befehl

```
> removePart X-4:A
Error, the BOM X does not contain the part A in the specified amount: 4
> removePart X-1:A
OK
> removePart X-2:A
OK
> removePart X-1:A
Error, the BOM X does not contain the specified part: A
```

Der `quit`-Befehl

Der parameterlose Befehl ermöglicht es, das laufende Programm vollständig zu beenden. Beachten Sie, dass hierfür keine Methoden wie `System.exit()` oder `Runtime.exit()` verwendet werden dürfen.

Eingabeformat

```
quit
```

Ausgabeformat

Im Erfolgsfall findet keine Ausgabe statt. Im Fehlerfall (z.B. bei einem falsch spezifizierten Eingabeformat) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

Beispielablauf: `quit`-Befehl

```
> quit quit
Error, incorrect input format, the quit command does not accept any parameters: quit
> quit
```

Beispielablauf

Beispielablauf: Teil 1 von 2

```
> addAssembly A=10:B;12:C
OK
> printAssembly a
Error, no BOM exists in the system for the specified name: a
> printAssembly A
B:10;C:12
> addAssembly B=18:E;15:D;27:F
OK
> printAssembly B
D:15;E:18;F:27
> addAssembly C=11:E;18:D;52:G
OK
> printAssembly C
D:18;E:11;G:52
> printAssembly D
COMPONENT
> getAssemblies A
C:12;B:10
> getAssemblies B
EMPTY
> getAssemblies D
Error, no BOM exists in the system for the specified name: D
> getComponents A
G:624;D:366;E:312;F:270
> getComponents B
F:27;E:18;D:15
> getComponents C
G:52;D:18;E:11
> getComponents D
Error, no BOM exists in the system for the specified name: D
> addAssembly D=9:F;12:G
OK
> printAssembly D
F:9;G:12
> getAssemblies A
D:366;C:12;B:10
> getAssemblies B
D:15
> getAssemblies C
D:18
> getAssemblies D
EMPTY
> getComponents A
G:5016;F:3564;E:312
> getComponents B
G:180;F:162;E:18
> getComponents C
G:268;F:162;E:11
> getComponents D
G:12;F:9
```

Beispielablauf: Teil 2 von 2

```
> addPart A+1:A
Error, adding the part A to the BOM A would create a cycle in the structure: A-A
> addPart D+1:A
Error, adding the part A to the BOM D would create a cycle in the structure: A-B-D-A
> addPart E+1:A
Error, no BOM exists in the system for the specified name: E
> addAssembly E=1:A
Error, the specified BOM E would create a cycle in the product structure: A-B-E-A
> removeAssembly B
OK
> removeAssembly C
OK
> printAssembly A
B:10;C:12
> getAssemblies A
EMPTY
> getComponents A
C:12;B:10
> printAssembly B
COMPONENTError, no BOM exists in the system for the specified name: B
> printAssembly C
COMPONENTError, no BOM exists in the system for the specified name: C
> printAssembly D
F:9;G:12
> printAssembly E
Error, no BOM exists in the system for the specified name: E
> addAssembly Abc=1906:iJk
Error, the amount for the component iJk in the BOM Abc is too high: 1906
> addAssembly Abc=19:iJk
OK
> printAssembly Abc
iJk:19
> addAssembly iJk=6:xyZ
OK
> printAssembly iJkxyZ
xyZ:6
> getAssemblies Abc
iJk:19
> getComponents Abc
xyZ:114
> addPart Abc+1:xyZ
OK
> printAssembly Abc
iJk:19;xyZ:1
> addPart iJk+1:xyZ
OK
> removePart iJk-7:xyZ
OK
> getAssemblies Abc
EMPTY
> getComponents Abc
iJk:19;xyZ:1
> quit
```