

252-0027-00: Einführung in die Programmierung

Übungsblatt 10 (Bonus)

Abgabe: 26 November 2024, 19:00

Checken Sie mit Eclipse wie bisher die neue Übungs-Vorlage aus. Importieren Sie das Eclipse-Projekt für die Bonusaufgabe.

Achtung: Diese Aufgabe gibt Bonuspunkte (siehe "Leistungskontrolle" im www.vvz.ethz.ch). Die Aufgabe muss eigenhändig und alleine gelöst werden. Die Abgabe erfolgt wie gewohnt per Push in Ihr Git-Repository auf dem ETH-Server. Verbindlich ist der letzte Push vor dem Abgabetermin. Auch wenn Sie vor der Deadline committen, aber nach der Deadline pushen, gilt dies als eine zu späte Abgabe. Bitte lesen Sie zusätzlich [die allgemeinen Regeln](#).

Aufgabe 1: ChronoCruiser-Factory (Bonus!)

Für diese Aufgabe sollen Sie ein Rechnungssystem für die Produktionslinie des ChronoCruiser 3000 entwickeln. Je nach Konfiguration des ChronoCruiser 3000 müssen verschiedene Fahrzeugkomponenten verbaut werden. Da verschiedene Komponenten jedoch durch verschiedene Zulieferungsunternehmen produziert werden, müssen sowohl deren Produktionskosten als auch die durch den Einbau entstehende Mehrwertsteuer berechnet werden.

Im bereitgestellten Projekt finden Sie drei Klassen `Cost`, `Part` und `Factory`.

`Cost` ist eine einfache Klasse, welche sowohl die Produktionskosten (Attribut `productionCost`), die Mehrwertsteuer ¹ (Attribut `vat`), als auch die Luxussteuer (Attribut `luxuryTax`, Teil (b)) eines Produkts speichert. Die Attribute sind alle vom Typ `int`.

`Part` ist eine Klasse mit einer Methode `process(Cost c)` und dient als Basis für verschiedene Unterklassen. Die Methode `process` modifiziert das gegebene `Cost`-Objekt so, dass die Produktionskosten und Steuern nach der Verarbeitung der Komponente in `c` hinterlegt sind. Jede dieser Unterklassen implementiert die Methode `process` so, dass die Produktionskosten und Steuern verschiedener Bauteile, wie in der folgenden Tabelle beschrieben, berechnet werden.

¹MWSt, "value added tax"

Komponente	Produktionskosten	MWSt
Flügeltüren	2000	3%
Fluxkompensator	Verdopplung der aktuellen Produktionskosten	7%
Schwebeumwandlung	Erhöhung um 20% der aktuellen Produktionskosten	10%
Einklappbare Räder	0 bis 7000: Erhöhung um 7000, aber maximal auf 100000	7%
Outatime-Kennzeichen	100 bis 50000: Erhöhung um 100, aber mind. auf 50000	10%

Factory enthält eine statische Methode `computeCost`. Diese Methode erhält ein Array von `Part`-Objekten und gibt ein `Cost`-Objekt zurück, welches die finalen Produktionskosten, die Mehrwertsteuer und (in Teil (b)) die Luxussteuer des Endprodukts speichert. Für die Berechnung dieses Objekts iteriert die Methode über die gegebenen `Part`-Objekte und ruft für jedes Objekt die Methode `process` auf. Die Luxussteuer ist nur relevant für Teil (b).

Berechnungsbeispiel Die Gesamtkosten für einen `ChronoCruiser` werden kumulativ berechnet und dann in einem `Cost`-Objekt gespeichert, abhängig von den verbauten Komponenten. Nehmen wir zum Beispiel an, dass die aktuellen Produktionskosten x und ein Mehrwertsteuerbetrag von t schon berechnet wurden. Wird nun eine Schwebeumwandlung verbaut, dann werden die Produktionskosten um 20% auf $1.2 \cdot x$ erhöht. Zudem wird eine Mehrwertsteuer auf die Differenz der neuen und alten Produktionskosten berechnet, also $(1.2 \cdot x - x) \cdot 10\%$ (siehe Mehrwertsteuersatz der Schwebeumwandlung). Der Gesamtbetrag der Mehrwertsteuer entsteht also aus diesem neuen Betrag und dem alten Mehrwertsteuerbetrag $t^{\text{neu}} = (1.2 \cdot x - x) \cdot 10\% + t$.

Sie sollen in allen Teilaufgaben **int-Arithmetik** für die Berechnungen verwenden. Für das eben genannte Berechnungsbeispiel könnte das wie folgt aussehen:

```
int tnew = ((120*x)/100 - x)*10/100 + t;
```

Aufgaben Bearbeiten Sie nun die folgenden Teilaufgaben:

- Kostenberechnung** Implementieren Sie die `process`-Methode für die gegebenen `Part`-Unterklassen aus der Tabelle oben, sodass die Produktionskosten und die Mehrwertsteuer der Komponente entsprechend der angegebenen Werte in `Cost` gespeichert werden. Zur Berechnung der Gesamtkosten wird dabei die Methode `Factory.computeCost` verwendet, welche bereits gegeben ist.
- Luxussteuern** Erweitern Sie Ihre Implementierung so, dass die Methode `computeCost` nach Berechnung der gesamten Kosten eine zusätzliche Luxussteuer in `Cost.luxuryTax` speichert. Die Luxussteuer wird nur erhoben, wenn mindestens eine der folgenden zwei neuen Komponenten verbaut wird:

Komponente	Produktionskosten	MWSt	Luxussteuer
First-Edition-Fluxkompensator	wie ein regulärer "Fluxkompensator"		5%
Verchromte Räder	wie reguläre "Einklappbare Räder"		5%

Die Luxussteuer berechnet sich als 5% der gesamten Produktionskosten des Endprodukts, falls mindestens eine der beiden Luxuskomponenten verbaut wurde. Beim Einbau mehrerer Luxuskomponenten wird trotzdem nur eine einmalige Luxussteuer erhoben.

- c) **Konfigurationsfehler** Passen Sie die Methode `computeCost` so an, dass `null` zurückgegeben wird, falls die Liste an Komponenten einen der folgenden Konfigurationsfehler aufweist:
- Es wurde mehr als ein Fluxkompensator verbaut.
 - Es wurde eine Schwebeumwandlung verbaut, nachdem bereits ein Fluxkompensator verbaut wurde.
 - Es wurde ein Outatime-Kennzeichen verbaut, nachdem bereits ein Fluxkompensator verbaut wurde.

Beachten Sie, dass diese Konfigurationsfehler die Luxuskomponenten aus Teil (b) nicht betreffen. Zum Beispiel ist es erlaubt, sowohl einen First-Edition-Fluxkompensator als auch einen regulären Fluxkompensator zu verbauen, nur nicht mehr als einen regulären Fluxkompensator.

Sie finden einige Testfälle für die jeweiligen Teilaufgaben in der Klasse `FactoryTest` im "test"-Ordner, welche Sie für eine erste Überprüfung Ihrer Implementierung verwenden können. Beachten Sie allerdings, dass die Testfälle nicht vollständig sind.