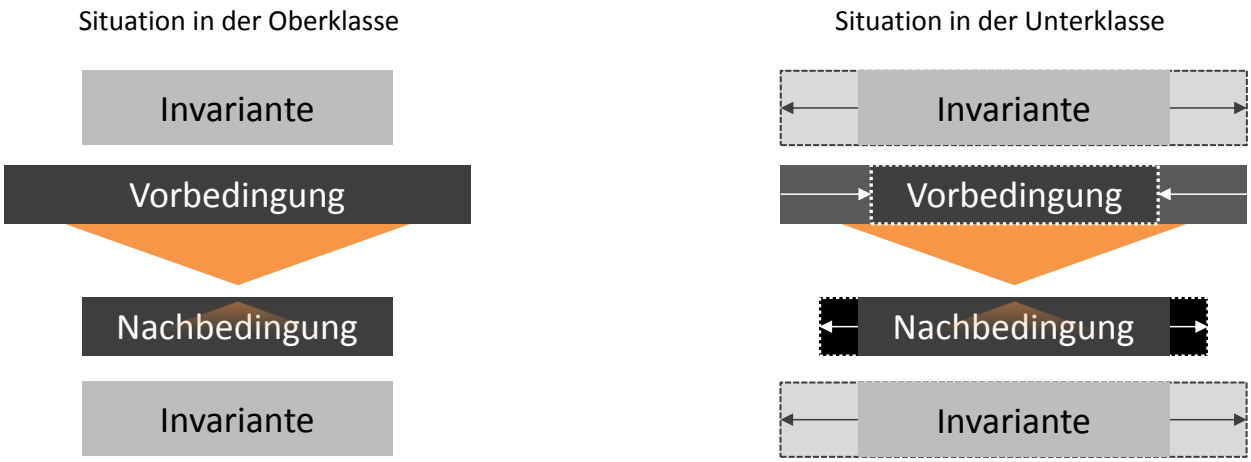
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN
WÜRZBURG-SCHWEINFURT

Programmieren II
Sommersemester 2016

Übungsblatt 5 (2 Punkte)

Liskovsches Substitutionsprinzip

Eine Spezialisierung im Sinne des Liskovschen Substitutionsprinzips liegt vor, wenn der Spezialfall alle Aufgaben der generelleren Lösung auch lösen kann – und noch etwas drüber hinaus. M.a.W.: alle Softwareverträge der Oberklasse müssen erfüllt werden, können aber auch „übererfüllt“ werden.



Konkret bedeutet das für alle öffentlichen Methoden der Oberklasse: Die Nachbedingungen müssen in der Unterklasse genauso erfüllt werden wie in der Oberklasse (ggf. sogar noch etwas mehr), die Vorbedingungen der Oberklasse müssen für die Ausführung der Methode in der Unterklasse ausreichen (ggf. kann diese sogar noch etwas weniger fordern) und die durch die Methode in der Oberklasse nicht veränderten Attribute und Aussagen bleiben auch in der Unterklasse unverändert (ggf. bleibt sogar noch etwas mehr unverändert).

Aufgabe „exercise05“ (Abgabe vom 26.4 - 2.5.2016, bZv-relevant)

An einer anderen (!) Fakultät häufen sich Beschwerden, dass der Eis-Automat in der Cafeteria zwar den richtigen Betrag Wechselgeld zurückgibt, dieser aber oft aus übermäßig vielen kleinen Münzen besteht. Die Kollegen wenden sich jetzt an uns von der Fakultät Informatik/Wirtschaftsinformatik, da der Hersteller des Eis-Automaten auf Nachfrage folgende Auskunft gegeben hat:

> *Der bei Ihnen aufgestellte Eis-Automat ist ein Gerät der neusten Generation. Im Gegensatz zu früheren Geräten kann jede Funktion des Automaten mit Hilfe von Java konfiguriert werden. Bei der Geldrückgabe heißt das konkret: wir liefern nur einen Vorschlag für die Geldrückgabelogik aus, die Sie Ihren Bedürfnissen anpassen können!*

Unsere Standardimplementierung finden Sie in der Klasse `SimpleChangeCalculator`. Darin enthalten ist eine Methode `getChange`, die zu einem vorgegebenen Geldrückgabebetrag die Aufteilung in Münzen errechnet. Dazu wird ein `int`-Array zurückgeliefert, das die Anzahl der einzelnen Münztypen wie folgt enthält:

An der Position 0: Anzahl der 1-Cent Münzen
An der Position 1: Anzahl der 2-Cent Münzen
An der Position 2: Anzahl der 5-Cent Münzen
An der Position 3: Anzahl der 10-Cent Münzen
An der Position 4: Anzahl der 20-Cent Münzen
An der Position 5: Anzahl der 50-Cent Münzen
An der Position 6: Anzahl der 1-Euro Münzen
An der Position 7: Anzahl der 2-Euro Münzen

Wir sichern Ihnen zu, dass der Rückgabebetrag richtig ist. Für die Stückelung der Münzen haben wir auf einen einfachen und bewährten Algorithmus zurückgegriffen.

Wenn Sie eine veränderte Stückelungslogik realisieren möchten, haben wir dies bereits für Sie vorbereitet: In der Klasse `EnhancedChangeCalculator` (einer Unterklasse von `SimpleChangeCalculator`) können Sie die Methode `getChange` überschreiben. Natürlich muss weiterhin sichergestellt sein, dass der richtige Betrag zurückgegeben wird (Liskovsches Substitutionsprinzip). Allerdings können Sie selbst bestimmen, welche Münzen genau zurückgegeben werden.

Sie können sich vorstellen, dass diese Auskunft bei der anderen Fakultät erst einmal große Verwirrung gestiftet hat.

Ihre Aufgabe:

Implementieren Sie im Package `exercise05` in der Klasse `EnhancedChangeCalculator` eine verbesserte Logik für die Stückelung der Geldrückgabe, indem Sie die Methode `getChange` überschreiben. Dabei soll jeweils ein `int`-Array mit der minimalen Anzahl an Münzen zurückgegeben werden, die den geforderten Betrag ergeben.

Im Verzeichnis Übungsaufgabe_5 finden Sie dazu eine Hilfklasse `Coin`, welche die zur Verfügung stehenden Münztypen repräsentiert. Außerdem ist die Klasse `SimpleChangeCalculator` und die Klasse `EnhancedChangeCalculator` enthalten, deren `main`-Methode einen groben Test durchführt.