Laborprüfung

PRP1 WI1 WiSe 2015/16 – 27.01.2016 – Axel Schmolitzky

Nachname, Vorname:		
Matrikelnummer:		Erfolgreiche Testfälle (nur vom Prüfer auszufüllen):
Account-Name:		

Oben in der Tabelle bitte nur Namen, Matrikelnummer und Account-Namen eintragen!

BlueJ unbedingt umstellen auf die deutsche Oberfläche, falls nicht bereits geschehen.

Die Prüfung besteht aus vier Teilaufgaben. Es empfiehlt sich, die Aufgaben in der genannten Reihenfolge zu bearbeiten. Zu jeder Aufgabe gibt es eine Testklasse, die euch Feedback darüber gibt, ob eure Implementation vollständig ist. Die Implementationen werden auch bewertet, wenn sie nur einen Teil der Tests erfolgreich bestehen. Es ist also möglicherweise besser, zur nächsten Aufgabe zu wechseln, wenn eine Teilaufgabe im Detail zu viel Zeit kostet.

Die Testklassen liegen nicht im Quelltext vor, lassen sich also nicht neu übersetzen. Solange ihr die Interfaces nicht verändert, ist das kein Problem. Solltet ihr dies aus Versehen tun, einfach BlueJ beenden und neu starten. Dann sind wieder alle Testklassen ausführbar.

Sorgt bei euren selbst geschriebenen Klassen dafür, dass sie übersetzbar sind - sonst werden sie nicht bewertet.

Aufgabe 1: ServiceMix

Setze als erstes Deinen Namen und Deine Matrikelnummer hinter das @author-Tag.

Implementiere die Rümpfe der Methoden in der Klasse ServiceMixImpl.

In dieser Klasse muss nicht auf null-Parameter geprüft werden.

Die Methode enthaeltZiffer soll rekursiv bleiben und lediglich so umgeschrieben werden, dass sie nur eine return-Anweisung enthält.

Die Methode nurVokale soll rekursiv implementiert werden.

Aufgabe 2: Geldbörse

Setze Deinen Namen und Deine Matrikelnummer hinter das @author-Tag.

Implementiere eine Klasse GeldboerseImpl, die das Interface Geldboerse implementiert. Unbedingt darauf achten, dass die neue Klasse genauso heißt wie hier vorgegeben. Die Klasse muss einen parameterlosen Konstruktor anbieten, der public ist.

Im Doku-Ordner stehen Folien zu Enums in Java.

Die Operationen zum Einstecken sind leichter zu implementieren als die zum Entnehmen, genauso wie die Operationen ohne Sammlungen als Parameter im Vergleich zu denen die mit Sammlungen als Parameter.

Die Methode geldPassendVorhanden ist anspruchsvoll - nicht zu viel Zeit mit der Implementierung verbringen und lieber zur nächsten Aufgabe!

Die Testklasse enthält Positiv- und Negativtests. Wenn explizit spezifiziert ist, dass eine Exception geworfen werden soll, wird dies auch überprüft.

Aufgabe 3: Züge

Setze Deinen Namen und Deine Matrikelnummer hinter das @author-Tag.

In dieser Aufgabe geht es um Züge, die aus *Zugelementen* (einer Lokomotive und beliebig vielen Waggons) bestehen. Unter einem *Zugteil* wird im Folgenden ein beliebiger zusammenhängender Teil eines Zuges verstanden, der aus mindestens einem Zugelement besteht.

Implementiere eine Klasse Lokomotive, die das Interface KannEtwasAnhaengen implementiert. Die Klasse muss einen parameterlosen Konstruktor anbieten, der public ist.

Implementiere eine Klasse Waggon, die die beiden Interfaces KannEtwasAnhaengen und Anhaengbar implementiert. Ein Waggon soll bei seiner Erzeugung die Anzahl der Passagiere erhalten, die in genau diesem Waggon sitzen.

Unbedingt darauf achten, dass die neuen Klassen genauso heißen wie hier vorgegeben.

Wenn beide Klassen ihre Interfaces korrekt implementieren, kann für einen Zug (bestehend aus einer Lok und beliebig vielen Waggons) die Anzahl der Zugelemente und der Passagiere im gesamten Zug ermittelt werden.

Aufgabe 4: ServiceMix2

Setze Deinen Namen und Deine Matrikelnummer hinter das @author-Tag.

Implementiere das Interface ServiceMix2 mit einer Klasse ServiceMix2Impl. Die Klasse muss einen parameterlosen Konstruktor anbieten, der public ist.

Unbedingt darauf achten, dass die neue Klasse genauso heißt wie hier vorgegeben.