Laborprüfung

PRP WI1 WiSe 2017/18 – 26.01.2018 – 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 umstellen auf die **deutsche Oberfläche**, falls nicht bereits geschehen.

Die Prüfung besteht aus **vier Teilaufgaben**. Zu jeder Aufgabe gibt es eine Testklasse, die Feedback darüber gibt, ob eine 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 Du die Interfaces nicht veränderst, ist das kein Problem. Solltest Du dies aus Versehen tun, einfach BlueJ beenden und neu starten. Dann sind wieder alle Testklassen ausführbar.

Sorge bei allen von Dir bearbeiteten Klassen dafür, dass sie übersetzbar sind – sonst werden sie nicht bewertet!

Lies Dir für jede Aufgabe gründlich die folgenden **Hinweise** UND das zugehörige **Interface** durch.

Aufgabe 1: ServiceMix implementieren

Implementiere die Rümpfe der Methoden in der Klasse ***ServiceMixImpl***.

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

Die Methode ***anzahlAuftreten*** soll **rekursiv** implementiert werden.

In dieser Klasse braucht generell **nicht** auf null-Parameter geprüft werden.

Falls Du **Hilfsmethoden** implementierst, müssen diese **private** deklariert sein.

Aufgabe 2: TicTacToe-Spielfeld mit Maps implementieren

Implementiere eine Klasse ***SpielfeldGeflecht***, die das Interface ***Spielfeld*** implementiert. Achte unbedingt darauf, dass die neue Klasse genauso heißt wie hier vorgegeben.

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

Die Klasse **muss** einen parameterlosen Konstruktor anbieten, der public ist.

In der Implementation **muss** die vorgegebene Klasse ***SpielfeldZeile*** verwendet werden. Außer ***SpielfeldZeile*** darf **im ersten Schritt** keine andere Klasse verwendet werden. Eine **vollständige** Lösung muss intern eine Map verwenden, die von einer gegebenen Zeilennummer auf eine ***SpielfeldZeile*** abbildet. Tipp: ***SpielfeldZeile*** verwendet intern auch eine Map.

Verändere im zweiten Schritt die vorgegebene Klasse ***SpielfeldZeile*** so, dass sie statt eines int-Wertes für den Spieler eine Referenz auf ein Element eines Aufzählungstyps ***Spieler*** liefert bzw. bekommt. Dieser Aufzählungstyp soll von Dir geeignet definiert und in der Schnittstelle von ***SpielfeldZeile*** (und somit auch in ***SpielfeldGeflecht***) verwendet werden. Das Interface ***Spielfeld*** soll aber unverändert bleiben!

Im Doku-Ordner stehen Folien zu **Enums** in Java.

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

Aufgabe 3: SanduhrFinder implementieren

Implementiere eine Klasse ***SanduhrFinderImpl***, die das Interface ***SanduhrFinder*** implementiert. Achte darauf, dass die neue Klasse genauso heißt wie hier vorgegeben.

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

Die Klasse muss einen parameterlosen Konstruktor anbieten, der public ist.

Da die Orientierung bei zweidimensionalen Arrays nicht eindeutig ist: Alle Tests gehen davon aus, dass der übergebene Parameter ein **Array von Zeilen** ist.

Ein Tipp: zweidimensionale Arrays in Java müssen nicht immer „rechteckig“ sein.

Diese Aufgabe ist schwieriger als Aufgabe 4, also bei Problemen erstmal die vierte Aufgabe probieren.

Aufgabe 4: Invertieren einer verketteten Liste

Implementiere in der Klasse ***LinkedWortListe*** die Methode ***invertieren***, so dass die Reihenfolge der enthaltenen Elemente umgedreht wird. Eine **vollständige** Lösung darf keine neuen Knoten erzeugen.

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