Übung 05

Korrektur: Sebastian Pritz

Name: Theresa Gschweidl

Punkte: 88

Allgemeine Anmerkungen:

1 Tag verspätet abgegeben -4

Beispiel 1: Flugreisen 22 von 30

Lösungsidee: 3/3

Anmerkung	**Abzug**
Passt!	

Quellcode: 11/17

Anmerkung	**Abzug**
Warum wird in keinem der Anwendungen der Konstruktor zur Initialisierung verwendet? Bei Person wäre setPerson(...) 1:1 Fall für den Konstruktor – bei den anderen Klassen gibt es nicht einmal Initialisierungsfunktionen und alle Attribute sind public. Das ist nicht nur mehr Schreibarbeit, sondern verletzt auch die Regeln der objektorientierten Programmierung (Datenkapselung). In diesem Fall könnte man, mit Ausnahme der Überladung, eigentlich structs statt Klassen verwenden.	**-5**
Bei << von AirFlight sollten alle Informationen, d.h. auch die Passagiere, ausgegeben werden. (wie man es als Mitarbeiter in einem Reisebüro erwarten würde)	**-1**
Sonst aber ganz okay 😊	

Testfälle: 8/10

Anmerkung	**Abzug**
Keine Flugreise ohne Personen und/oder Flüge	**-2**

Beispiel 2: Stücklistenverwaltung 70 von 70

Lösungsidee: 7/7

Anmerkung	**Abzug**
Um auf die Unsicherheit in der Lösungsidee zu antworten: Mit delete bzw. im Destruktor löscht du Pointer auf Objekte, die du in der Klasse abspeicherst, aber nicht selbst die Klasse sind. Man verwaltet ja in CompositePart eine Liste von anderen Parts → drüber iterieren und diese löschen. Wenn du das Objekt löscht, in dessen Funktion du dich gerade befindest, geht das natürlich schief 😊	Hinweis

Quellcode: 40/40

Anmerkung	**Abzug**
Normalerweise sollte man Instanzen mit dem Schlüsselwort „new“ anlegen. Das heißt „Part p* = new Part(...)“, und nicht mit „Part p(...)“. Zweiteres existiert nur innerhalb der momentanen Funktion (erstes potentielles Problem) und der Speicher kann nicht händisch freigegeben werden, wie oben beschrieben. Dadurch, dass es zu keinen Fehlern geführt hat, unterlasse ich aber die Abzüge. Gut implementiert und kommentiert! 😊	**Hinweis**

Testfälle: 23/23

Anmerkung	**Abzug**
Passt, sehr gut 😊	