

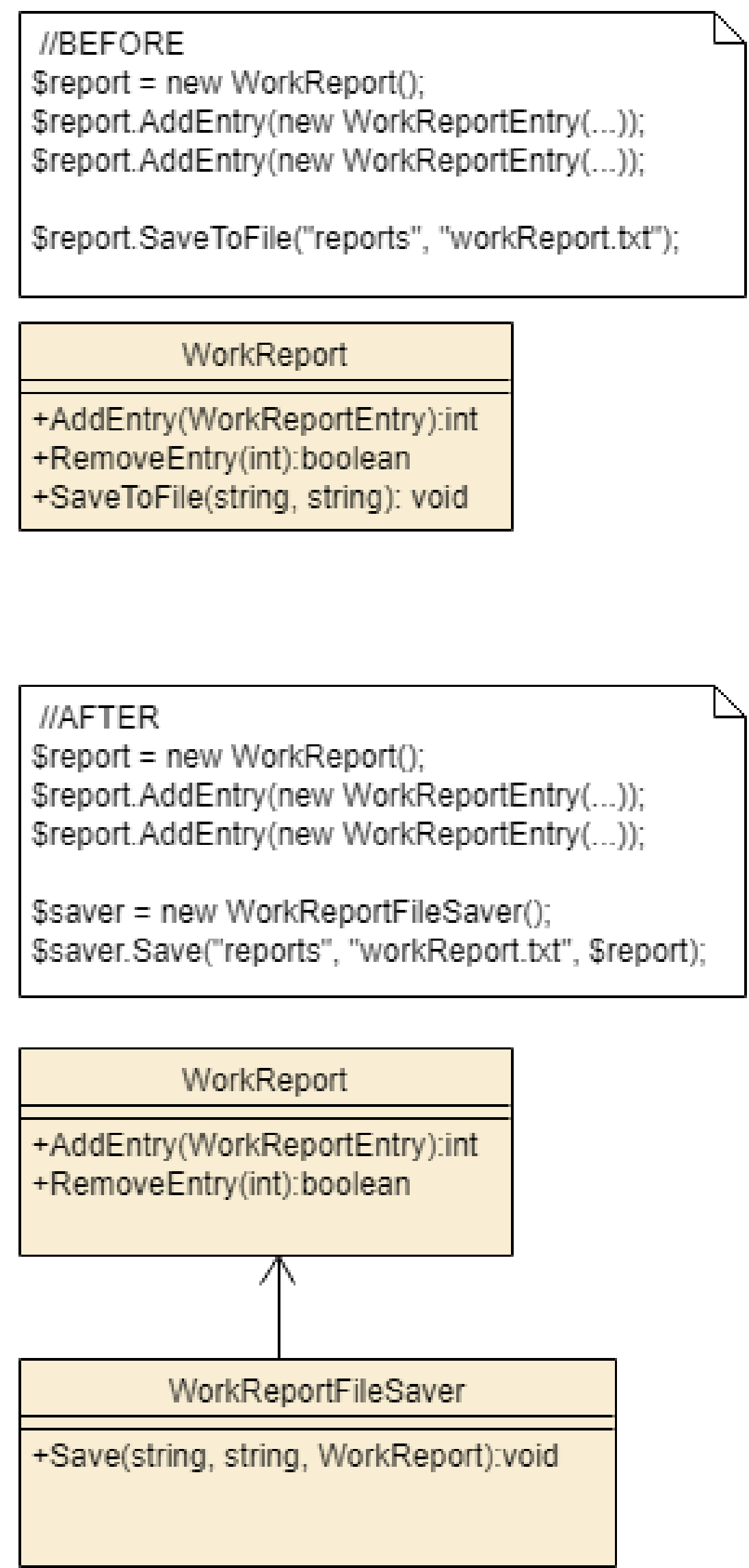
Single-Responsibility

Es sollte niemals mehr als einen Grund geben, eine Klasse zu verändern.

Viele kleine Klassen sind besser als wenige große.

Umsetzung um Fehler zu vermeiden und eine klare Strukturierung zu bekommen.

Annahme: Es sollen Reporte geschrieben und als Files gespeichert werden.

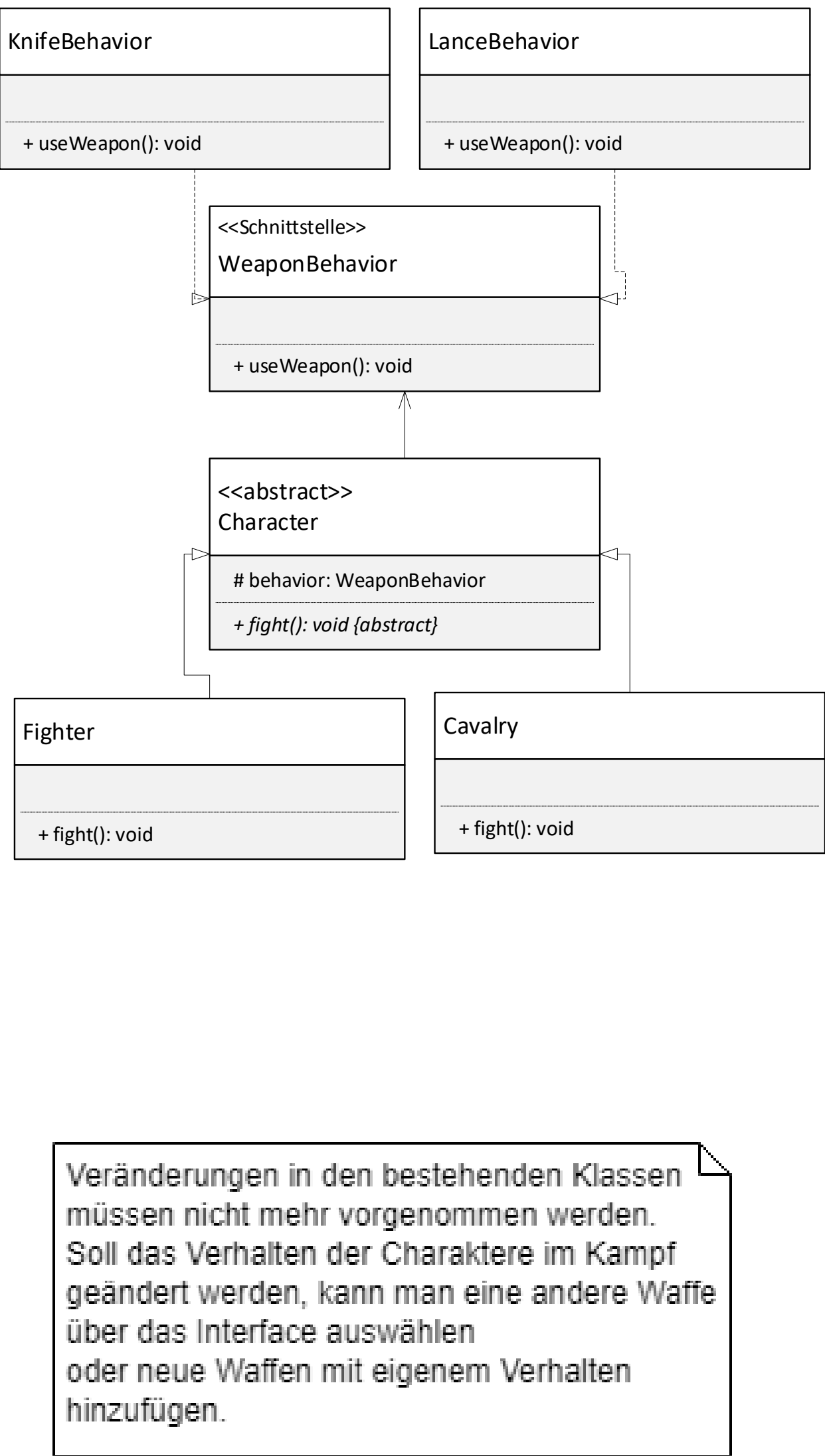


Open-Closed

Eine Klasse soll offen für Erweiterung, aber geschlossen für Veränderung sein.

Umsetzung um Fehler in fertig implementierten Klassen zu vermeiden.

Annahme: Es gibt verschiedene Charaktere, welche eine Waffe besitzen. Die Waffen haben eine Funktion zum Ausführen, welche je nach Waffe ein anderes Verhalten hat.

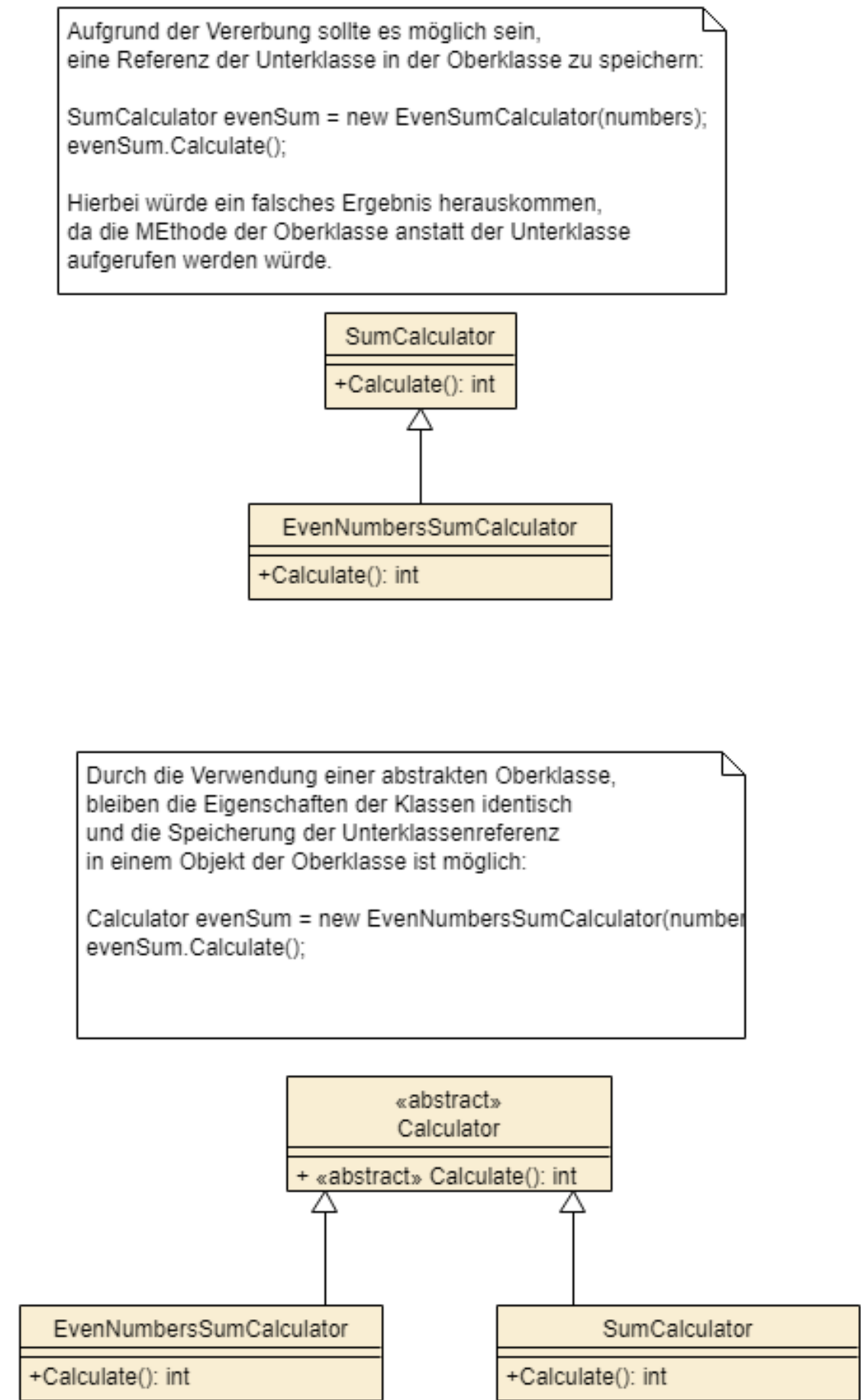


Liskov-Substitution

Eine Unterklasse muss immer alle Eigenschaften der Oberklasse erfüllen und darf sie nicht verändern.

Abgeleitete Klassen müssen immer anstelle der Basisklasse einsetzbar sein.

Annahme: Es sollen Werte zusammengerechnet werden. Weiter sollen die dafür geltenden Werte z.B. nach geraden Zahlen gefiltert werden.

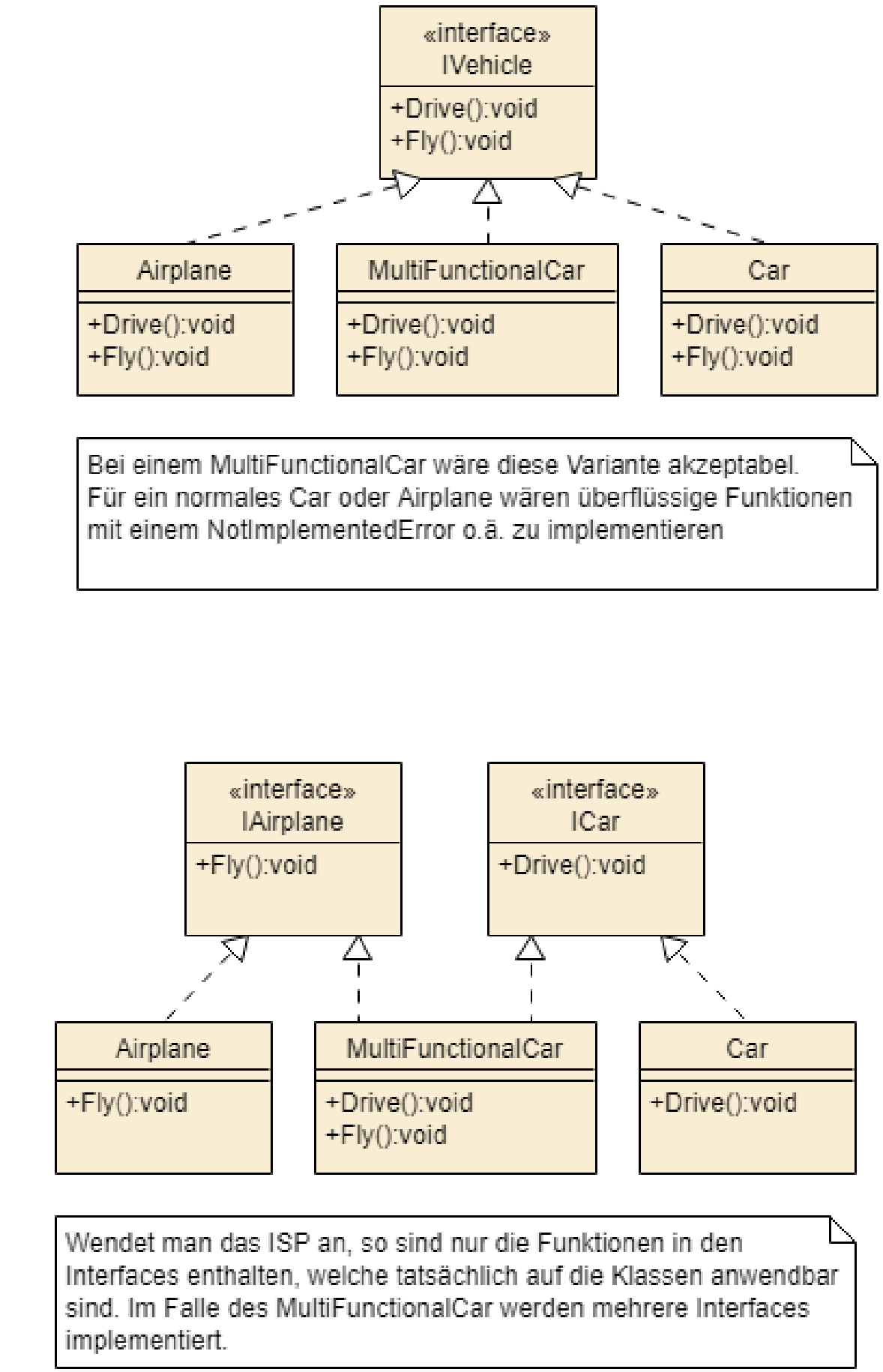


Interface-Segregation

Zu große Interfaces sollen in mehrere aufgeteilt werden.

Ein Interface soll nur Funktionen enthalten, die eng zusammengehören.

Annahme: Verschiedene Transportmittel die Fliegen und/oder Fahren können.



Dependancy-Inversion

Hohe Module sollen nicht von niedrigeren Modulen abhängig sein.

Stabile Abstraktionen > flüchtige Konkretionen.

Annahme: Es existiert eine Lampe, welche mit dem Stromnetz verbunden sein soll.

