# Kurzbaufsatz der objektorientieren Entwurfsprinzipien lt. Galileo-Buch Kapitel 3

## Prinzip einer einzigen Verantwortung (Single Responsibility Principle im SOLID)

Jede Klasse bzw. auch jede Methode soll genau eine Aufgabe erldedigen. Also die Klasse Währungsrechner soll beispielsweise Methoden für die Umrechnung bieten aber nicht Methoden für das Logging.

* Wartbarkeit ist somit gegeben
* Ist im ganzen Projekt vorhanden

## Trennung der Anliegen - ähnlich wie das Prinzip einer einzigen Verantwortung (Seperation of Concerns im SOLID)

* Jede Funktionalität soll separat implementiert werden
* Der Code bleibt erweiterbar (man muss dann nicht mehrere Klassen abändern)
* Ist im ganzen Projekt vorhanden

## Wiederholungen vermeiden

Redundanz also Codduplikate müssen vermieden werden. Das Programm sollte auch in einzelne Module zerlegt werden. Durch das Aufteilen in viele Module kann man jene Module, welche eine Aufgabe erledigen, die man mehrmals benötigt, leichter wiederverwenden.

Ebenso kann man mithilfe von Template Hook Codeduplikate vermeiden.

## Offen für Erweiterung, geschlossen für Änderung (Open-Closed-Principle)

Der Code sollte offen für Erweiterungen sein. Das bedeutet, dass man Änderungen vornehmen kann, ohne dass man andere Klassen umschreiben muss.

## Trennung der Schnittstelle von der Implementierung

Dies ermöglicht es Module einfach auszutauschen welche die Schnittstellen implementieren, so kann einfach Code ersetzt oder abgeändert werden, ohne großartig den restlichen Code zu berühren.

Wichtig hierbei ist, dass die Schnittstellen und die einzelnen Module exakt eine Aufgabe erledigen. Das bedeutet, dass die Schnittstellen möglichst klein gehalten werden sollen.

## Umkehr der Abhängigkeiten (Dependency Inversion im SOLID)

Die Idee von der Umkehr der Abhängigkeiten ist es, die direkte Abhängigkeit zwischen den Modulen aufzuheben und dadurch die Wiederverwendbarkeit zu erhöhen. Dies geschieht mit der Verwendung eines Interfaces also einer Schnittstelle zwischen den Modulen automatisch.

Umkehrung des Kontrollflusses (Hollywoodprinzip = Don’t call us, we call you)

## Mach es Testbar

Das Letzte Entwurfsprinzip lt. Galileo-Buch Kapitel 3 erklärt, dass man darauf achten sollte, dass man alle Module bzw. Methoden der Software unabhängig voneinander testen sollte.

Um dies Ordnungsgemäß umzusetzen, sollte man parallel zum Coden JUNIT-Tests für jede Methode schreiben.