Gehen Sie in einem Kurzaufsatz auf alle objektorientieren Entwurfsprinzipien lt. Galileo-Buch Kapitel 3 (<http://openbook.rheinwerk-verlag.de/oop/oop_kapitel_03_001.htm#mja6c3f342e440956adade4ec377b02c1a>) sowie auf die SOLID-Prinzipien (soweit nicht überschneidend mit den Galileo-Buch-Prinzipien) ein. Dazu ist das jeweilige Prinzip in eigenen Worten nochmals zu beschreiben und dessen Anwendung ist im erstellten Code an möglichst vielen Stellen zu illustrieren.

**SOLID Prinzipien**

**Single Responsibillity Prinzip**

Dieses Prinzip sagt aus, dass jede Klasse nur eine einzige Verantwortung / Aufgabe haben soll. Dadurch wird die Notwendigkeit einer späteren Änderung verringert.

In diesem Projekt kann man dieses Prinzip gut an den einzelnen, tatsächlichen Währungsrechnern erkennen. Deren einzige Aufgabe ist es, den Betrag mit dem entsprechenden Kurs zu multiplizieren.

**Open Closed Prinzip**

Erweiterungen sollen möglich sein ohne das Grundlegende Verhalten zu ändern.   
Dieses Prinzip trifft zu da es relativ einfach möglich ist, einen neuen Währungsrechner hinzu zu fügen. Man muss nur dessen Namen und den entsprechenden Kurs auswählen. Da die Restlichen Festlegungen bereits vom abstrakten Währungsrechner getroffen werden.

**Liskovsches Substitutionsprinzip**

Eine Instanz einer Sub Klasse muss sich so verhalten wie man es von er Super Klasse erwartet. Ist ersichtlich am Beispiel der Währungsrechner man erwartet von dem Rechner das er einen gegeben Wert in einen anderen umwandelt. Dies erfüllen sowohl der Euro2Dollar als auch der Euro2Yen Rechner auf Ihre bestimmte weise.

**Interface Segregation Prinzip**

Kein Client soll dazu gezwungen sein, Methoden zu verwenden die er nicht braucht. Zu große Schnittstellen sollen in mehrere Kleine Schnittstellen aufgeteilt werden. In diesem Projekt sieht man es am Interface IUmrechnen das wiederum von abstrakten Klassen implementiert wird die verschiedene Zusatzfunktionen bereitstellen.

**Dependency-Inversion-Prinzip**

Module einer hohen Rangordnung sollen nicht von Methoden mit niedriger Rangordnung abhängen. Umso weiter oben man sich in der Hierarchie befindet umso allgemeiner sind die Funktionen definiert. In unserem Projekt können wir es am Beispiel der Umrechnung sehen. Das Interface gibt nur die Funktionen vor der Abstrakte Währungsrechner stellt Zusatz Funktionen bereit und erst die eigentlichen Calculator führen die Berechnung durch.