ADAP CW#07

# Projektdaten

Projekt name: 3D-Printing

Projekt repository: [https://github.com/**ModischFabrications**/wahlzeit](https://github.com/ModischFabrications/wahlzeit)

This week’s tag: adap-hw07 on master

Homework diff: <https://github.com/ModischFabrications/wahlzeit/compare/adap-hw06...adap-hw07>

CI: [https://travis-ci.org/**ModischFabrications**/wahlzeit](https://travis-ci.org/ModischFabrications/wahlzeit)

Docker Hub Repo: [https://hub.docker.com/r/**modischfabrications**/wahlzeit](https://hub.docker.com/r/modischfabrications/wahlzeit)

# Hausaufgaben

## Implementation

### Neue Klassen (falls vorhanden)

### Veränderte Klassen (falls vorhanden)

### Erklärung

Ich habe damit angefangen im Interface eine abstrakte Methode “assertClassInvariants” zu definieren, die in den Kindklassen implementieren wurden. Hierbei werden die Werte auf NaNs geprüft und im Fall von den SphericCoordinates auch auf valide Wertebereiche, der Aufruf der Prüfung kann durch diese Hooks schon in der Oberklasse stattfinden. Diese Invarianten werden in allen public-Methoden von AbstractCoordinate verwendet, um inkonsistente Zustände noch vor Auslieferung an den Nutzer abzufangen. Genauso wurden die Rückgaben der Konvertierungsmethoden abgesichert. Dabei sind vor allem die Invarianten relevant, da der Zustandsraum der Koordinatenklasse nur sehr eingeschränkt ist.

Gleichheitsprüfungen und hashCodes wurden absichtlich ausgelassen, da hier die Konsistenz utilitaristisch ist und nichts zur Prüfung selbst beiträgt.

Die Parameter und die Instanz selbst wurden nicht auf die Preconditions geprüft, da dies nach “Design-by-Contract” Aufgabe des Nutzers ist.

Add design-by-contract to Coordinate interface and class hierarchy

–Use assert statements and assertion methods for both

●Preconditions

●Postconditions

Zuletzt habe ich alle Änderungen gepusht und den letzten Commit mit dem oben genannten Tag für die Abgabe markiert.

## Fragen

### How do you test for conditions?

Ich habe die Java-eigenen assert-Statements genutzt und (wo sinnvoll) durch Kommentarstrings ergänzt. Problem bei der großzügigen Verteilung von Prüfungen ist eine Verschlechterung der Sigal-to-Noise-Ratio, da der funktionale Code zunehmend von Prüfungen versteckt wird, daher wurden Prüfungen möglichst hoch in der Klassenhierarchie ausgeführt.

### Would you rather use Java’s assert or a dedicated assertion method? Or something else?

Ich habe überwiegend die Java-eigenen assert-Statements genutzt und würde das auch wieder tun bei so kleinen Prüfungen. Komplexe Sachverhalte würde ich dennoch in eine eigene Methode ausgliedern und eventuell sogar mit einer eigenen Exception kombinieren, da hierdurch die Lesbarkeit durch kleinere Überlappung von nicht-funktionalen Testcode und Implementation verbessert wird.