ADAP CW#07

Projektdaten

Projekt name:	3D-Printing
Projekt repository:	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
Docker Hub Repo:	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.

In dessen Implementation 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 Invariantenprüfungen werden in allen public-Methoden von AbstractCoordinate ausgeführt, um inkonsistente Zustände noch vor Auslieferung an den Nutzer abzufangen. Genauso wurden die Rückgaben der Konvertierungsmethoden abgesichert.

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 dann auf die Preconditions geprüft, Fehler wurden nicht versucht zu korrigieren, da dies nach "Design-by-Contract" Aufgabe des Nutzers ist. Da alle Parameter Koordianten sind werden auch bei den Preconditions die Invarianten geprüft. Andere Prüfungen sind nicht relevant, da der Zustandsraum der Koordinatenklasse nur sehr eingeschränkt ist und andere fehlerhafte Zustände nicht möglich sind.

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, da diese bei kleinen Einzeilern den geringsten Overhead haben. Komplexe Sachverhalte wie in Invarianz würde ich 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. Sonderlösungen mit externen Libraries würde ich nur mit sehr gutter Begründung nutzen, da hier die Abhängigkeiten des Projekts unnötig aufgebläht werden.