ADAP CW#06

Projektdaten

Projekt name:	3D-Printing
Projekt repository:	https://github.com/**ModischFabrications**/wahlzeit
This week's tag:	adap-hw06 on master
Homework diff:	https://github.com/ModischFabrications/wahlzeit/compare/adap-hw05...adap-hw06
CI:	https://travis-ci.org/**ModischFabrications**/wahlzeit
Docker Hub Repo:	https://hub.docker.com/r/**modischfabrications**/wahlzeit

Hausaufgaben

Implementation

Neue Klassen (falls vorhanden)

AbstractCoordinate

Veränderte Klassen (falls vorhanden)

(div durch subpackaging)

CartesianCoordinate, SphericCoordinate,

CartesianCoordinateTest, SphericCoordinateTest

Erklärung

Angefangen habe ich mit dem Aufräumen des bisherigen Standes, dabei habe ich Subpackages eingeführt, um die Struktur des Projekts zu verfeinern und die Lesbarkeit und das Scoping zu verbessern.

Ich bin bei den in den alten Klassen verwendeten Attributen geblieben und bin nicht auf die neue Notation mit Lat/Longitude gewechselt, um die bereits bestehende Logik der Koordinatenklasse nicht erneut anpassen zu müssen. Dadurch wird die Arbeit und Fehlerrate bei gleicher Funktionalität minimiert.

Dann habe ich nach und nach die Implementierung des Interfaces in die abstrakte Elternklasse gezogen, die konkreten "do*" Algorithmen habe ich dabei an die implementierten Kindklassen delegiert. Für die Gleichheit habe ich komplett auf die CartesianCoordinate verwiesen, Kindklassen können dennoch eine eigene, performantere Version der Gleichheitsprüfung implementieren und die Methode überschreiben.

Die Tests mussten fast nicht angepasst werden, da bereits in der letzten Hausaufgabe alle relevanten Testfälle abgedeckt wurden. In den Tests für die Kindklassen wird explizit auf die do* Methode

verwiesen, um die Fehlerstreuung durch das Wandeln zwischen den Systemen zu minimieren.

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

Fragen

How did you decide which methods go into the abstract superclass or in the implementations?

Ich habe mich dazu entschieden die komplexen Konvertierungsalgorithmen in den Implementierungen zu lassen und deren Aufrufe dahin zu delegieren, um die abstrakte Klasse möglichst High-Level zu halten und nur Redundanzen zu minimieren. Gleichheitsprüfungen wurden nur einmal in Cartesian implementiert und als default dahin delegiert, könnten zugunsten der Performance aber auch erneut für Spheric implementiert werden. So wird das Interface vollständig in der Abstract erfüllt und die Kindklassen bestehen vor allem aus primitives.

similar() wurde als Utility-Funktion ebenfalls in die Abstract gezogen, würde thematisch aber noch besser in eine generische Math-Library passen.