**Model**

Das Model beheimatet alle Daten sowie ein Grossteil der Algorithmen (zusätzliche Algorithmen sind in der Klasse Calc des Package Utilities zu finden). Das Model ist observable um ein update() aus der View zu ermöglichen.

*Model*

Die Klasse Model erzeugt im Konstruktor für jede Berechnungsart einen closedLoop. Sie enthält setter- und getter-Methoden zur Verarbeitung der Daten in weiteren Klassen.

*ClosedLoop*

In der Methode closedLoop wird über eine Case-Abfrage die jeweilige Berechnungsart der passende Controller erstellt. Zusätzlich beinhaltet die Klasse die Methode calculate(). Diese berechnet zum einen die Schrittantwort mittels calculateStepResponse() und für die Phasengangmethode zusätzlich die overShootOptimazation().

*Controller*

Der Controller bildet die Oberklasse aller Faustformeln und der Phasengang-Methode. Er beinhaltet die abstrakte Klasse calculate() sowie alle nötigen setter- und getter- Methoden.

Faustformeln

*Chien20*

Die Klasse Chien20 stellt die Algorithmen zur Berechnung der Reglerwerte gemäss Chien/Hrones/Reswick (20%) zur Verfügung und erbt von Controller.

ChienApper, Oppelt, Rosenberg und ZiglerNicholas sind die Klassen zu den zugehörigen Fausformeln. In diesen giebt es jeweils eine Methode calculate() in der die benötigten Werte aus dem Path ausgelesen werden und die Berechnungen mittels vorgegebenen Fausformeln, je nach Controllertyp (PID,PI), durchgefürt wird.

PhaseRespsonseMethod

In dieser Klasse wird die OmegaAchse in Abhängigkeit vom Phasengang und des benötigten Winkelbereiches erstellt.

Ausserdem finden wir verschiedene calculate Merhoden.

Einerseits ist da calculate() , in dieser wird die UTF Strecke aus der Strecke(path) geholt und die Omega-Achse Methode aufgeruffen. In dieser wird Sie Abhängigkeit vom Phasengang und des benötigten Winkelbereiches erstellt.

Anschlissend werden die Werte für Hs und phiS berehnet.

Andererseits sind da die Methoden, calculateTnvTnk() um Tnk und Tnv zu berechnen, calculateKrk um Krk zu berechnen, calculatecontrollerConf() um eine umrechung von Bodekonformen Werten zu Reglerkonformen Werten in der calc Klasse auszulösen, calculatePhaseMargin() bestimmt je nach Reglertyp den Phasenrand und zum schluss calculateOverShoot(), hier wird je nach vorher berechnetem Überschwingen, dem Wert phiu einen der 4 vordefinierten static final Werte zugewiesen.

Path

Hier giebt es ebenfalls eine calculate() Methode die mittels der Sani Methode aus dem Calc die Strecke ausrechnet.

Die restlichen Klassen sind reine Getr/Setr Klassen.

Die Alghorythmen zu diesen Methoden finden Sie im Anhang.