**Model**

Die Model-Klassen beheimaten alle Daten sowie ein Grossteil der Algorithmen (zusätzliche Algorithmen, die von den Model-Klassen verwendet werden, sind in der Klasse Calc des Package Utilities zu finden). Das Model ist observable um das Aktualisieren der View zu ermöglichen.

*Model*

Wie im Klassendiagramm ersichtlich erzeugt die Klasse Model im Konstruktor für jede Berechnungsart eine Instanz der Klasse ClosedLoop. Das Model hat ein Objekt der Klasse Path und übergibt dieses an den ClosedLoop. Das Model enthält Setter- und Getter-Methoden zur Verarbeitung der Daten mittels weiterer Klassen. Das Model nutzt die Möglichkeit notifyObservers() aufzurufen, um update() auf der View zu bewirken.

*ClosedLoop*

Die Instanzen der Klasse ClosedLoop repräsentieren die geschlossenen Regelkreise. Sie kennen ihren Berechnungstyp und besitzen einen Regler.

Im Konstruktor wird je nach Berechnungsart ein passender Controller instanziiert. Zusätzlich beinhaltet die Klasse die Methode calculate(). Diese berechnet für alle Berechnungsarten die Schrittantwort mittels calculateStepResponse() und für die Phasengang-Methode zusätzlich die overShootOptimazation(). Zudem sind diverse Setter- und Getter-Methoden Bestandteil dieser Model-Klasse.

*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 um Werte zu setzten und auszulesen.

*Chien20*

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

*ChienApper*

Die Klasse ChienApper stellt die Algorithmen zur Berechnung der Reglerwerte gemäss der Faustformel Chien/Hrones/Reswick (apperiodisch) zur Verfügung erbt von Controller, instanziiert und setzt eine Übertragungsfunktion.

*Oppelt*

Die Klasse Oppelt stellt die Algorithmen zur Berechnung der Reglerwerte gemäss der Faustformel Oppelt zur Verfügung, erbt von Controller, instanziiert und setzt eine Übertragungsfunktion.

*Rosenberg*

Die Klasse Rosenberg stellt die Algorithmen zur Berechnung der Reglerwerte gemäss der Faustformel Rosenberg zur Verfügung, erbt von Controller, instanziiert und setzt eine Übertragungsfunktion.

*PhaseResponseMethod*

Die Klasse PhaseResponseMethod stellt die Algorithmen zur Berechnung der Reglerwerte gemäss der Phasengang-Methode zur Verfügung.

Sie erbt von der Klasse Controller und bringt all derer Setter- und Getter-Methoden mit.

Über die überladene Methode setData() werden die Input-Werte gesetzt. Dabei wird die Methode calculateOverShoot(), die das für die Berechnung des korrekten Überschwingens benötigt Attribut phiU setzt, ausgelöst. Zusätzlich wird calculate() aufgerufen. calculate() berechnet anhand der Methode createOmegaAxis() die diskrete Omega-Achse in Abhängigkeit der Zeitkonstante der Regelstrecke. Die Übertragungsfunktion in s der Regelstrecke wird für alle Punkte von Omega berechnet. calculateTnk() wird ausgelöst und berechnet Tnk und Tvk unter Zuhilfenahme der diskreten Werte. calculateTnk() löst wiederum calculateKrk() zur Berechnung von Krk aus und ruft zudem calculateControllerConf() und setUTF() auf. calculateControllerConf() transformiert die Werte in die reglerkonforme Darstellung. setUTF() setzt die Übertragungs-Funktion des Reglers.

*Path*

Die Klasse Path berechnet in der Methode calculate(), unter Zuhilfenahme der Sani-Methode (Calc.sani()), die Strecke (Path).

Besitzt eine Instanz der Klasse UTF sowie Setter- und Getter-Methoden zum Setzen und Auslesen von Werten.

*UTF*

Setzt und speichert die Übertragungsfunktion. Diese kann über diverse Getter- und Setter-Methoden ausgelesen bzw. geschrieben werden.