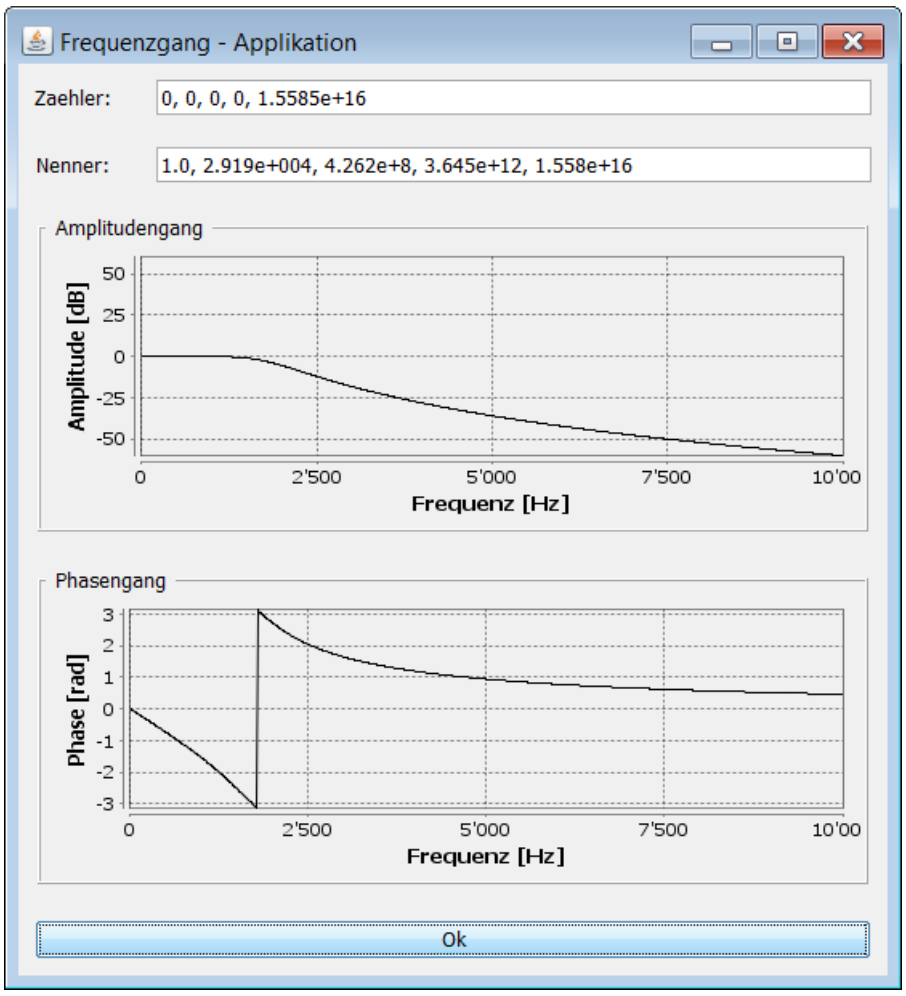
Übung Frequenzgang

Ziel dieser Übung ist es, eine Applikation zur Berechnung und Darstellung von Frequenzgängen zu Programmieren. Die Applikation soll dabei folgende Erscheinung haben:



Der Frequenzgang sei wie in Matlab üblich, wie folgt gegeben:

$$H(\omega) = \frac{b_0 \cdot (j \cdot \omega)^N + b_1 \cdot (j \cdot \omega)^{N-1} + b_2 \cdot (j \cdot \omega)^{N-2} + \ldots + b_N}{a_0 \cdot (j \cdot \omega)^M + a_1 \cdot (j \cdot \omega)^{M-1} + a_2 \cdot (j \cdot \omega)^{M-2} + \ldots + a_M}.$$

Die Information bezüglich der Koeffizienten des Zähler- und Nennerpolynoms wird dabei mittels der beiden Textfelder eingegeben. Die Eingabe erfolgt in Form von durch Leerzeichen getrennten Zahlen. Die Klassen zur Darstellung von Amplituden- und Phasengang sind bereits gegeben. Sie verwenden zur Darstellung der Daten das Package ***jfreechart*** (http://www.jfree.org/jfreechart/). Die Applikation basiert auf dem Model-View-Controller Pattern.

Aufgabe 1: Erstellen des Grundgerüstes

Als erstes wollen wir das Grundgerüst des Programms anhand des nachfolgenden Klassendiagramms erstellen. Ein paar Elemente der Klassen sind dabei bereits gegeben. Die Klassen sind in einzelnen Dateien zum implementieren.

a) Erstellen Sie sämtliche Klassen-, Attribut- und Methodendeklarationen gemäss Klassendiagramm.
b) Implementieren Sie die fehlenden Interfaces mit zugehörigen Methoden.
c) Versehen Sie jede Methode mit einer kurzen Beschreibung. Fragen Sie bei Unklarheiten den Dozenten.

Aufgabe 2: Erstellen der Klasse View

Das User-Interface ist als ***GridBagLayout*** mit 5 Zeilen und 2 Spalten organisiert. In x-Richtung wachsen die Textfelder, die beiden Plots sowie der OK-Button mit, in y-Richtung wachsen nur die beiden Plots mit. Die Klassen ***Amplitudengang*** und ***Phasengang*** sind bereits vorgegeben und müssen nur noch instanziert werden.

a) Erstellen Sie das User-Interface gemäss diesen Angaben.
b) Implementieren Sie die restlichen Methoden der Klasse ***View***.
c) Testen Sie ihren Code.

Aufgabe 3: Erstellen der Klasse Controller

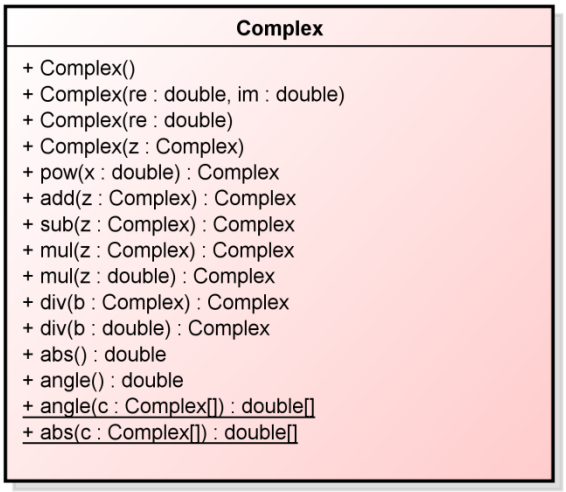
Der Kontroller agiert als Schaltzentrale bezüglich User-Interaktion. Mit dazu gehört die Aufgabe, aus den Zeichenketten in den Textfeldern double - Arrays zu erzeugen.

a) Implementieren Sie die Methoden der Klasse ***Controller***.
b) Überlegen Sie sich eine einfache Art um ***stringToCoeff***() zu testen.

Aufgabe 4: Erstellen der Klasse Complex

Der Klasse ***Complex*** kommt der Umgang mit komplexen Zahlen zu. Java kann, wie die meisten Sprachen, nicht per se komplex Rechnen.

a) Schreiben Sie eine Klasse ***Complex*** um die benötigten, komplexen Rechenoperationen, sowie den zu einer komplexen Zahl zugehörigen Real- und Imaginärteil zu kapseln.
b) Überlegen Sie, wie sie die Methoden testen könnten.



Aufgabe 5: Erstellen der Klasse PicoMatlab

Die Klasse ***PicoMatlab*** beinhaltet die statische Methode ***freqs(double[] b, double[] a, double[] f)*** und ist aus didaktischen Gründen vorhanden. ***freqs***() berechnet aufgrund der Arrays mit den Koeffizienten des Zähler- und Nennerpolynoms den komplexwertigen Frequenzgang.

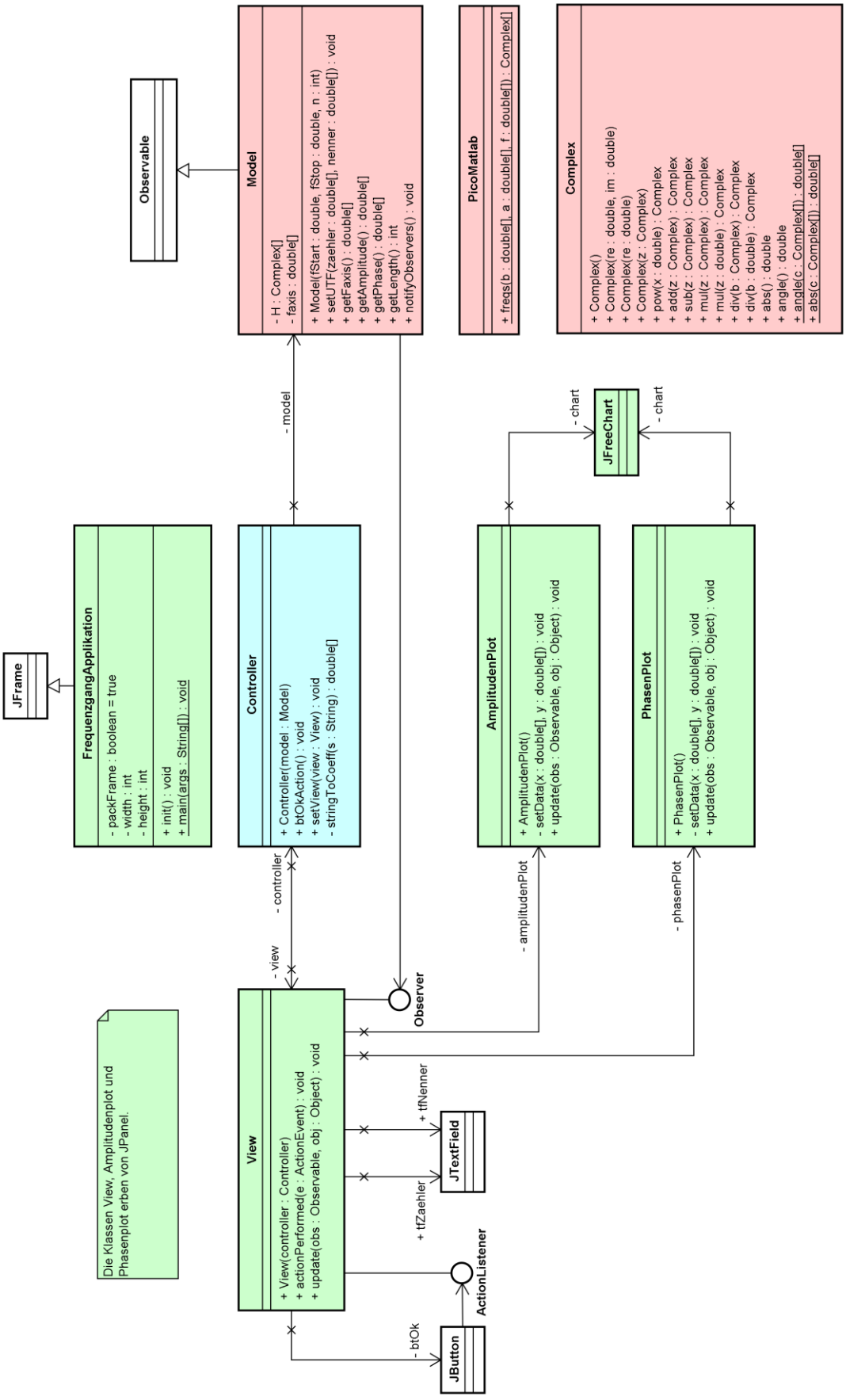
a) Implementieren Sie die Methode ***freqs***().
b) Überlegen Sie sich eine einfache Art um die Methode zu testen.

Aufgabe 6: Erstellen der Klasse Model

Die Klasse Model beheimatet die Daten des Frequenzganges, nämlich die Frequenzachse und den komplexwertigen Frequenzgang. Mit den Methoden ***getFaxis***(), ***getAmplitude***() und ***getPhase***() kann die Information ausgelesen werden. Mit ***setUTF***() wird eine neue Übertragungsfunktion festgelegt.

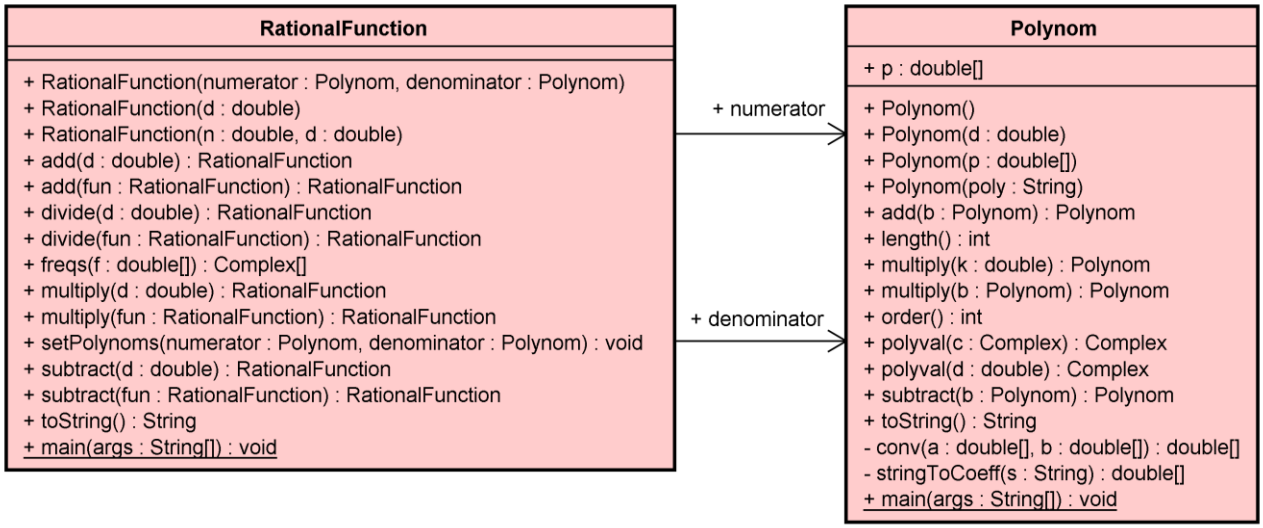
a) Implementieren Sie die Methoden der Klasse ***Model***.
b) Überlegen Sie sich eine einfache Art um die Methoden zu testen.

Klassendiagramm:



Erweiterung des Projektes um die Klassen Polynom und RationalFunction

Wir wollen das Projekt nun um die Klassen Polynom und RationalFuntion erweitern. Die Klasse *Polynom* kapselt die Koeffizienten des *Polynoms* und beinhaltet die mit einem *Polynom* einhergehenden Methoden. Die Klasse *RationalFunction* kapselt die Polynome der gebrochenrationalen Funktion und beinhaltet die mit einer solchen Funktion einhergehenden Methoden:



Aufgabe 7: Erstellen der Klassen Polynom und RationalFunction

Teile der Klasse *Polynom* und der Klasse *RationalFunction* wurden bereits im Unterricht geschrieben. Die Klassen sollen nun noch komplettiert und getestet werde.

a) Implementieren Sie die verbleibenden Methoden der Klasse *Polynom*. Testen Sie anhand einiger Handbeispiele die Korrektheit der Berechnungen.
b) Implementieren Sie die verbleibenden Methoden der Klasse *RationalFunction*. Testen Sie anhand einiger Handbeispiele die Korrektheit der Berechnungen.

Aufgabe 8: Modifikation der Berechnung

Die Berechnungen sollen nun aufgrund der nachfolgenden Struktur gebaut werden. Das Model besitzt nun eine *RationalFunction,* die wiederum das Zähler- und Nennerpolynom beinhaltet.

Klassendiagramm V2Plus: