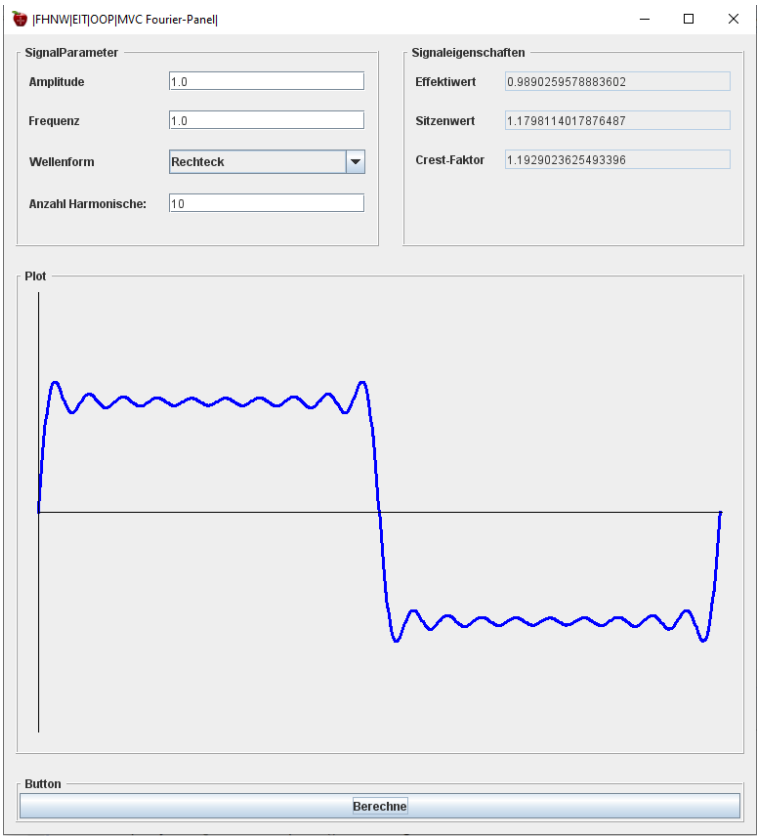
Übung: MVC Fourier Rechner

Ziel dieser Übung ist es, den einfache Fourier-Rechner aus OOP1 ins MVC-Entwurfsmuster zu konvertieren. Er berechnet aufgrund der Amplitude, der Frequenz, der Wellenform und Anzahl der Harmonischen das Fourier Signal sowie dessen charakteristische Grössen.



Der Rechner ist im Model-View-Controller Entwurfsmuster gehalten. Die **View** verfügt über die Dateneingabe und über die Repräsentation des Resultates. Dem **Controller** obliegt die Steuerung der Applikation, er delegiert die Berechnung ans **Model**. Das Model macht die benötigten Berechnungen und notifiziert allfällig registrierte **Observer**.

Die **View** ist zum genannten Zwecke in vier Panel gegliedert: Das *ParameterPanel* beheimatet die benötigten Eingabetextfelder sowie die Auswahl. Das *PropertyPanel* zeigt die charakteristischen Grössen des Signales. Dem *PlotPanel* (in der Vorlage bereits vorgegeben) kommt die grafische Repräsentation des Signalverlaufes zu. Das *ButtonPanel* verfügt über den Button *Berechne* und löst damit, via den **Controller**, die Berechnung aus. Die Berechnung kann auch direkt via Eingabe-Taste in einem Textfeld oder durch die Auswahl ausgelöst werden. Der **Controller** liest auf das entsprechende Ereignis hin die Werte aus den Textfeldern aus und delegiert die Berechnung ans Model.

Das Model berechnet die Signalform aufgrund der zugehörigen Fourier-Reihe und bestimmt die charakteristischen Grössen. Es löst nach der Berechnung mittels **Observer-Observable Mechanismus** ein Aufdatieren der repräsentierten Daten aus.

Aufgaben :

a) Überlegen Sie sich die Funktion der einzelnen Klassen und Methoden im Kontext des MVC Entwurfsmusters. Das Grundgerüst des Klassendiagramms ist in der Vorlage bereits implementiert. Weitere Hinweise finden Sie auch als Pseudocode.

b) Implementieren Sie zunächst die Klasse **View** und implementieren Sie dann alle **Panels**.

c) Nun können Sie die Steuerung in der Klasse Controller realisieren. Dies funktioniert bisher völlig unabhängig davon, was das Modell am Ende berechnet.

d) Zum Schluss müssen wir noch die Klasse Model beleben. Hierzu können Sie einige Teile aus der FourierPanel Übung aus OOP1 verwenden.

Hinweis:
- Effektivwert (rms)

$$rms = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} y[n]^2}$$

- Spitzenwert: Absoluter Maximalwert

- Crest-Faktor (Scheitelfaktor)

$$crest = \frac{peak}{rms}$$

Das MVC Entwurfsmuster lässt sich schön anhand der Trace ersehen

Trace beim Aufstarten:

```
Start via main(String args[])
	Attribute von FourierPanelFrame@939047783 werden initialisiert ...
	Attribute von Model@548246552 werden initialisiert ...
	Konstruktor Model():Model@548246552 wird ausgeführt ...
	Attribute von Controller@835648992 werden initialisiert ...
	Konstruktor Controller():Controller@835648992 wird ausgeführt ...
	Attribute von View@492228202 werden initialisiert ...
	Konstruktor View():View@492228202 wird ausgeführt ...
		Attribute von ParameterPanel@401625763 werden initialisiert ...
		Konstruktor ParameterPanel():ParameterPanel@401625763 wird ausgeführt ...
		Attribute von PropertyPanel@501263526 werden initialisiert ...
		Konstruktor PropertyPanel():PropertyPanel@501263526 wird ausgeführt ...
		Attribute von PlotPanel@1190900417 werden initialisiert ...
		Konstruktor PlotPanel():PlotPanel@1190900417 wird ausgeführt ...
		Attribute von ButtonPanel@1598924227 werden initialisiert ...
		Konstruktor ButtonPanel():ButtonPanel@1598924227 wird ausgeführt ...
		Methode Controller@835648992.setView() wird ausgeführt ...
	Methode PlotPanel@1190900417.paintComponent()(paintCount: 0 ) wird ausgeführt ...
```

Trace beim Event Berechne:

```
Methode ParameterPanel@401625763.actionPerformed() wird durch Ereignis ausgelöst ...
	Methode Controller@835648992.btBerechne() wird ausgeführt ...
		Methode Model@548246552.berechne() wird ausgeführt ...
		Methode Model@548246552.berechneRechteck() wird ausgeführt ...
		Methode Model@548246552.notifyObservers() wird ausgeführt ...

Methode View@492228202.update() wird ausgeführt ...
	Methode PropertyPanel@501263526.update() wird ausgeführt ...
		Methode Model@548246552.getRms() wird ausgeführt ...
		Methode Model@548246552.getPeak() wird ausgeführt ...
	Methode PlotPanel@1190900417.update() wird ausgeführt ...
		Methode Model@548246552.getSignal() wird ausgeführt ...
	Methode PlotPanel@1190900417.paintComponent()(paintCount: 1 ) wird ausgeführt ...
```

Klassendiagramm: