```
package impulsdemo.gui.goodies;

import java.awt.GridBagLayout;

public class PseudoCode {}

class View extends JPanel implements Observer {
  // 5
  private TraceV2 tr = new TraceV2(this);

  /**
   * <pre>
   * - Baut User-Interface gemäss Aufgabenstellung.
   * - Registriert die View als Beobachter beim Model.
   * </pre>
   */
  public View(Controller controller) {

  /**
   * <pre>
   * - Datiert entsprechende Komponenten auf.
   * </pre>
   *
   * @param obs
   * @param obj
   */
  public void update(Observable obs, Object obj) {
}

class ReiterPane extends JTabbedPane {
  // 7
  private TraceV2 tr = new TraceV2(this);

  /**
   * <pre>
   * - Setzt bevorzugte Grösse auf (65% von DrehzeigerDemoApplikation.appWidth, 100% von
DrehzeigerDemoApplikation.appHeight).
   * - Setzt die Border mit entsprechendes Beschriftung.
   * - Baut das GUI mit den drei Reitern gemäss Aufgabenstellung.
   *   Hinweis: Komponenten werden mittels der Methode addTab(String title, Component
component) hinzugefügt.
   * </pre>
   *
   */
  public ReiterPane() {

  /**
   * <pre>
   * - Ruft passendes update() der entsprechenden Komponenten auf.
   * </pre>
   *
   * @param obs
   * @param obj
   */
  public void update(Observable obs, Object obj) {
}

class ParameterPanel extends JPanel implements ActionListener {
  // 51
  private TraceV2 tr = new TraceV2(this);
```



```
  /**
   * <pre>
   * - Setzt bevorzugte Grösse auf (35% von DrehzeigerDemoApplikation.appWidth, 100% von
DrehzeigerDemoApplikation.appHeight).
   * - Setzt entsprechendes Attribut.
   * - Setzt die Border mit entsprechender Beschriftung.
   * - Baut das GUI gemäss Aufgabenstellung.
   * </pre>
   *
   * @param controller
   *            Referenz des Controllers.
   */
  public ParameterPanel(Controller controller) {

  /**
   * <pre>
   * - Falls Quelle des Ereignisses gleich cbLTIbel:
   *   - Bei den entsprechenden Komponenten setEnabled() mit wahr resp. unwahr aufrufen.
   *   - Bei btReset Methode doClick() auslösen.
   * - Falls Quelle des Ereignisses gleich cbLTI2Ord:
   *   - Bei den entsprechenden Komponenten setEnabled() mit wahr resp. unwahr aufrufen.
   *   - Bei btReset Methode doClick() auslösen.
   * - Falls (Quelle des Ereignisses gleich btReset) ODER (Quelle des Ereignisses gleich
btStart):
   *   - Falls cbLTIbel selektiert ist:
   *     - Entsprechende Methode des Controllers mit Text des Zaehler- und des
Nenner-Textfeldes aufrufen.
   *   - Falls cbLTI2Ord selektiert ist:
   *     - Entsprechende Methode des Controllers mit Parameter (String)
jcbLTIType.getSelectedItem() aufrufen.
   *   - btReset() des Controllers aufrufen.
   * - Falls Quelle des Ereignisses gleich btStart:
   *   - Entsprechende Methode des Controllers aufrufen.
   *   *
   * </pre>
   */
  public void actionPerformed(ActionEvent e) {
}

class Controller {
  // 5
  private TraceV2 tr = new TraceV2(this);

  /**
   * <pre>
   * - Setzt entsprechendes Attribut.
   * </pre>
   *
   * @param model
   *
   */
  public Controller(Model model) {

  /**
   * <pre>
   * - Ruft entsprechende Methode des Models auf.
   * </pre>
   *
   * @param systemType
   */
  public void setSystemType(String systemType) {
```

```
  /**
   * <pre>
   * - Ruft entsprechende Methode des Models mit zugehörigen Arrays auf.
   * </pre>
   *
   * @param stZaehler
   * @param stNenner
   */
  public void setParameter(String stZaehler, String stNenner) {

  /**
   * <pre>
   * - Ruft entsprechende Methode des Models auf.
   * </pre>
   *
   */
  public void btReset() {

  /**
   * <pre>
   * - Ruft entsprechende Methode des Models auf.
   * </pre>
   *
   */
  public void btStart() {
}

class Model extends Observable implements ActionListener {
  // 43
  private TraceV2 tr = new TraceV2(this);

  /**
   * <pre>
   * - Baut einen Timer mit Intervall 40 ms und this als ActionListener.
   * - Startet den Timer.
   * </pre>
   */
  public Model() {

  /**
   * <pre>
   * - Falls NICHT angehalten:
   *   - processing() aufrufen.
   *   - n inkrementieren.
   * </pre>
   */
  public void actionPerformed(ActionEvent e) {

  /**
   * <pre>
   * - Sortiert die komplexwertigen Pole im Attribut P in aufsteigender Reihenfolge so,
   *   dass an der Stelle [0] der kleinste Imaginärteil zu liegen kommt.
   * - Werden P[i+1] und P[i] getauscht, so werden auch die zugehörigen Residuen im
   *   Attribut R[i+1] und R[i] getauscht!
   * </pre>
   */
  public void bubbleSort() {

  /**
   * <pre>
```



```
   * - Berechnet alle Drehzeiger gemäss Aufgabenstellung.
   * - Notifiziert die Observer.
   * </pre>
   */
  public void processing() {

  /**
   * <pre>
   * - Setzt n gleich Null.
   * - Setzt das Attribut angehalten auf wahr.
   * - Erzeugt einen neuen drehZeiger-Array mit der Länge von R.
   * - Ruft die Methode processing auf.
   * </pre>
   */
  public void reset() {

  /**
   * <pre>
   * - switch (systemType)
   *   - Fall LHE:
   *     - Methode setUTF() mit Argumenten B und A_LHE aufrufen.
   *   - Fall JW und RHE sinngemäss.
   * </pre>
   *
   * @param systemType
   */
  public void setSystemType(String systemType) {

  /**
   * <pre>
   * - Mit den Argumenten B,A ein neues Objekt der Klasse Residue erzeugen.
   *   Das Objekt enthält nun die komplexwertigen Residuen R, die zugehörigen Pole P
   *   sowie den konstanten Term K.
   * - Falls die Länge von R gerade ist:
   *   Mittels der Matlab-Methode concat(Complex[] a, Complex[] b) eine
   *   komplexe Null an residue.R und residue.P anhängen und in den Attributten
   *   R und P ablegen. Hinweis: 'concationate' heisst 'zusammenfügen'!
   * - Sonst:
   *   - residue.R und residue.P direkt in den Attributten R und P ablegen.
   * - bubbleSort() aufrufen.
   * - reset() aufrufen.
   * </pre>
   *
   * @param B
   * @param A
   */
  public void setUTF(double[] B, double[] A) {
}
```