```java
package modem.gui;

import java.awt.GridBagConstraints;

// Ich bestaetige, dass ich diese Pruefung selbstaendig geloest habe.
// Ich weiss, dass bei Zuwiderhandlung die Note 1 erteilt wird.
//
// Name:
// Vorname:

public class View extends JPanel implements Observer {
    // 8
    private static final long serialVersionUID = 1L;
    private BDGPanel bdgPanel;
    public ParameterPanel parameterPanel;
    private ReiterPanel reiterPanel;

    /**
     * <pre>
     * - Baut das GUI gemäß Aufgabenstellung.
     * </pre>
     *
     * @param controller
     *
     */
    public View(Controller controller) {

    /**
     * <pre>
     * - Ruft entsprechende Methode update() des beheimateten Panels auf.
     * </pre>
     */
    public void update(Observable obs, Object obj) {

}
```

```java
package modem.gui;

import java.awt.GridBagConstraints;

// Ich bestaetige, dass ich diese Pruefung selbstaendig geloest habe.
// Ich weiss, dass bei Zuwiderhandlung die Note 1 erteilt wird.
//
// Name:
// Vorname:

public class ParameterPanel extends JPanel implements ActionListener {
    // 35
    static final long serialVersionUID = 1L;
    private Controller controller;

    public JComboBox<?> cbKanalLaenge = new JComboBox<Object>(new String[] { "100", "200", "300" });

    public JTextField tfAWGNAmplitude = new JTextField("0.05");
    public JTextField tfTonAmplitude = new JTextField("0.0");
    public JTextField tfTonFrequenz = new JTextField("5e6");
    public JTextField tfSchrittFF = new JTextField("1e-5");
    public JTextField tfSchrittFB = new JTextField("1e-4");

    private JButton btReset = new JButton("Reset Equalizer");

    /**
     * <pre>
     * - Baut das GUI gemäß Aufgabenstellung.
     * - Setzt allfällige Attribute.
     * - Setzt die bevorzugte Grösse auf 50% von AdaptiveModemDemo.appWidth und 45% von
AdaptiveModemDemo.appHeight
     * </pre>
     *
     * @param controller
     *
     */
    public ParameterPanel(Controller controller) {

    @Override
    /**
     * <pre>
     * - Falls Quelle des Ereignisses gleich btReset:
     *   - Entsprechende Methode des Controllers aufrufen.
     * - Sonst:
     *   - setParameter() des Controllers aufrufen.
     * </pre>
     */
    public void actionPerformed(ActionEvent e) {
        // 4
    }
}
```

```java
package modem.gui;

import java.awt.Dimension;

// Ich bestaetige, dass ich diese Pruefung selbstaendig geloest habe.
// Ich weiss, dass bei Zuwiderhandlung die Note 1 erteilt wird.
//
// Name:
// Vorname:

public class ReiterPanel extends JTabbedPane {
    // 5
    private static final long serialVersionUID = -7176405344277868872L;
    private ViewFrequenzBereich vFrequenzBereich = new ViewFrequenzBereich();
    private ViewZeitBereich vZeitBereich = new ViewZeitBereich();

    /**
     * <pre>
     * - Setzt die bevorzugte Dimension auf (100%, 60%) von
     *   AdaptiveModemDemo.appWidth resp. AdaptiveModemDemo.appHeight.
     * - Baut das GUI gemäß Aufgabenstellung.
     * </pre>
     */
    public ReiterPanel() {

    /**
     * <pre>
     * - Ruft entsprechende Methode update() der beheimateten Panel auf.
     * </pre>
     *
     * @param obs
     * @param obj
     */
    public void update(~~Observable~~ obs, Object obj) {
}
```

```java
package modem.gui;

import java.util.concurrent.Executors;

// Ich bestaetige, dass ich diese Pruefung selbstaendig geloest habe.
// Ich weiss, dass bei Zuwiderhandlung die Note 1 erteilt wird.
//
// Name:
// Vorname:

public class Controller implements Runnable {
    // 13
    private Model model;
    private View view;
    private ScheduledExecutorService service = Executors.newSingleThreadScheduledExecutor();

    /**
     * <pre>
     * - Setzt entsprechendes Attribut.
     * - Bewirkt mittels service.scheduleAtFixedRate(Runnable runnable, long initialDelay,
     *   long periode, TimeUnit TimeUnit.MILLISECONDS), dass die Methode run() nach einer
     *   Anfangsverzögerung von 1500 ms alle 100 ms ausgeführt wird.
     * </pre>
     *
     * @param model
     */
    public Controller(Model model) {

    /**
     * <pre>
     * - Erzeugt ein Wrapper-Objekt parameter der Klasse Parameter.
     * - Holt den Text aus den Textfeldern des ParameterPanel der View und Wandelt sie in double - Zahlen.
     * - Setzt die entsprechenden Attribute im Objekt parameter.
     * - Holt den SelectedIndex der ComboBox cbKanalLaenge des ParameterPanels der View und
     *   setzt das entsprechende Attribute im Objekt parameter.
     * - Ruft entsprechende Methode des Models auf.
     * </pre>
     *
     */
    public void setParameter() {

    /**
     * <pre>
     * - Setzt entsprechendes Attribut ;-).
     * </pre>
     *
     */
    public void resetFilter() {

    /**
     * <pre>
     * - Setzt entsprechendes Attribut ;-).
     * </pre>
     *
     * @param view
     */
    public void setView(View view) {

    public void run() {

}
```

```java
package modem.model;

import java.util.Observable;

//Ich bestaetige, dass ich diese Pruefung selbstaendig geloest habe.
//Ich weiss, dass bei Zuwiederhandlung die Note 1 erteilt wird.
//
//Name:
//Vorname:

public class Model extends Observable implements SymbolListener {
    // 33
    public Delay delay;
    private FIRFilter transmitFilter;
    private FIRFilter kanalFilter;
    private LMSFilter ffFilter;
    private LMSFilter fbFilter;

    protected SymbolQuelle symbolSource = new SymbolQuelle(16 * 1024, this);
    private double[] fAxis = new double[1024];
    private double[] tAxis = new double[512];

    private Random random = new Random();

    private double schrittFF = 1e-5, schrittFB = 1e-4, fbOut = 0.0;
    private double tonFrequenz = 5e6;
    private double fb = 50e6, fs = 2 * fb, Ts = 1 / fs;
    private long n = 0;

    private double awgnAmplitude = 0.0;
    private double tonAmplitude = 0.0;

    /**
     * <pre>
     * - Erzeugt mit den Argumenten (16 * 1024, this) die SymbolQuelle.
     * - Erzeugt die Verzögerungsleitung delay mit einer Verzögerung von 32.
     * - Erzeugt das Transmit-Filter und das Kanal-Filter mit den Koeffizienten
     *   TxFlt.TxFltBCoeffs und Kanal.BCoeffs[0].
     * - Erzeugt die adaptiven LMS-Filter ffFilter und fbFilter mit den Längen 64 und 16.
     * - Erzeugt mittels linspace() von Matlab die fAxis und die tAxis entsprechender
     *   Längen und den Bereichen von Null bis fs/2 resp. von 0 bis Länge der tAxis minus 1.
     * </pre>
     */
    public Model() {

    public double[] getfAxis() {

    public double[] getImpulsFBFilter() {

    public double[] getImpulsFFFilter() {

    public double[] getImpulsKanalFilter() {

    public double[] getImpulsTransmitFilter() {

    public double[] gettAxis() {

    public void processSymbol(double[] symbol) {

    /**
     * <pre>
     * - Erzeugt ein neues ffFilter und fbFilter ursprünglicher Länge
     * </pre>
     */
    public void resetFilter() {

    /**
     * <pre>
```

```
     * - Erzeugt ein neues Kanal-Filter mit entsprechenden Kanal.BCoeffs[Index] Koeffizienten.
     * - Setzt die Attribute entsprechend den Werten im Objekt parameter.
     * </pre>
     *
     * @param parameter
     */
    public void setParameter(Parameter parameter) {

    public void notifyObservers() {
}
```

```java
package modem.model;

import java.util.concurrent.Executors;

//Ich bestaetige, dass ich diese Pruefung selbstaendig geloest habe.
//Ich weiss, dass bei Zuwiederhandlung die Note 1 erteilt wird.
//
//Name:
//Vorname:

public class SymbolQuelle implements Runnable {
    // 8
    private double[] symbol;
    private SymbolListener symbolListener;
    private ScheduledExecutorService service = Executors.newSingleThreadScheduledExecutor();

    /**
     * <pre>
     * - Erzeugt den double-Array symbol mit entsprechender Anzahl Elementen.
     * - Setzt das entsprechende Attribut.
     * - Bewirkt mittels service.scheduleAtFixedRate(Runnable runnable, long initialDelay,
     *   long periode, TimeUnit TimeUnit.MILLISECONDS), dass die Methode run() nach einer
     *   Anfangsverzögerung von 1500 ms alle 200 ms ausgeführt wird.
     * </pre>
     *
     * @param anzahl
     * @param symbolListener
     */
    public SymbolQuelle(int anzahl, SymbolListener symbolListener) {

    public void run() {
}
```