```java
package de.unitrier.st.fp.s23.blatt01;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


public class FunctionCurve extends Application
{
    public FunctionCurve() { }

    @Override
    public void start(Stage stage)
    {
        int width = 670;
        int height = 520;
        FlowPane p = new FlowPane();

        Scene scene = new Scene(p, width, height);

        Button btnReset = new Button("Reset"); //wichtig!! Button muss der Klasse javafx.scene.control angehören
        Button btnDraw = new Button("Draw");

        CheckBox cbColor = new CheckBox("Green"); // wichtig!! Button muss der Klasse javafx.scene.control angehören

        Slider lineWidth = new Slider();



        Spinner<Double> s1 = new Spinner(); //<Double> damit z.b. s1.getValue() unten funktionert
        Spinner<Double> s2 = new Spinner();
        Spinner<Double> s3 = new Spinner();
        Spinner<Double> s4 = new Spinner();

        s1.setValueFactory(new SpinnerValueFactory.DoubleSpinnerValueFactory(Double.MIN_VALUE,Double.MAX_VALUE,1,0.5));
//alle vier rufen die Methode
        s2.setValueFactory(new SpinnerValueFactory.DoubleSpinnerValueFactory(Double.MIN_VALUE,Double.MAX_VALUE,1,0.5));
//vom Spinner auf(=setValueFactory)
        s3.setValueFactory(new SpinnerValueFactory.DoubleSpinnerValueFactory(Double.MIN_VALUE,Double.MAX_VALUE,1,0.5));
//und erstellen im Parameter direkt
        s4.setValueFactory(new SpinnerValueFactory.DoubleSpinnerValueFactory(Double.MIN_VALUE,Double.MAX_VALUE,1,0.5));
//ein SpinnerValueFactory
        //die ersten zwei Werte sind obere und untere Schranken, viereter Paramter ist der Anfangsswert und der dritte
ist die Schrittweite


        FunctionCanvas canvas = new FunctionCanvas();

        canvas.setWidth(width);
        canvas.setHeight(height);
        canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue()); //getvalue ist eine
Objektmethode der Klasse Spinner und kriegt die Zahl dadrin
        //als Double. Funktionert nur wenn der Spinner als Liste deklariert is vom Typ Double. Siehe oben

        btnReset.setOnAction(e -> {
            s1.getValueFactory().setValue(1.0); //So kann den Wert des Spinners umändern über .getValueFactory().
setValue(0.0)
            s2.getValueFactory().setValue(1.0); //Der Parameter braucht analog zu getValue() ein Double Wert
            s3.getValueFactory().setValue(1.0);
            s4.getValueFactory().setValue(1.0);


            canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue());
            canvas.setGraphColor(Color.BLACK);
            canvas.drawFunctionCurve();

        });

        btnDraw.setOnAction(e -> {
            canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue());
            canvas.drawFunctionCurve();
        });


        //Ab hier kommen die Listener. Wichtig ist, dass Listener oft mit ...Property arbeitet. Das heißt er taucht
meistens oft
        //wenn davor eine Methode ...Property aufruft. Wie hier unten(s1.valueProperty().addListener...)

        s1.valueProperty().addListener(e-> {
            canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue());
            canvas.drawFunctionCurve();
        });
        s2.valueProperty().addListener(e-> {
            canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue());
            canvas.drawFunctionCurve();
        });
```

```
 82         s3.valueProperty().addListener(e-> {
 83             canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue());
 84             canvas.drawFunctionCurve();
 85         });
 86         s4.valueProperty().addListener(e-> {
 87             canvas.setFunctionParameters(s1.getValue(),s2.getValue(),s3.getValue(),s4.getValue());
 88             canvas.drawFunctionCurve();
 89         });
 90 
 91         stage.widthProperty().addListener(e -> { //Wichtig: Wie oben bei den Listenern von den Spinnern braucht man
    hier auch iwas Prop erty.
 92             canvas.setWidth(stage.getWidth());   //Man kann am Fenster selber (stage) ein Lisenter setzen der die
    Fensetergröße anpasst
 93             canvas.drawFunctionCurve(); //sobald sich das Fenster verändert. Entsprechend der Fenstergröße dann
 94         });stage.heightProperty().addListener(e -> {
 95             canvas.setHeight(stage.getHeight());
 96             canvas.drawFunctionCurve();
 97         });
 98 
 99         cbColor.setOnAction(e -> {  //Checkbox (nochmal: Der Klasse javafx.scene.control kann setOnAction annehmen
100 
101             if(cbColor.isSelected()) { //Merken: isSelected!!
102                 canvas.setGraphColor(Color.GREEN);
103             }
104             else {
105                 canvas.setGraphColor(Color.BLACK);
106 
107             }
108             canvas.drawFunctionCurve();
109         });
110 
111         lineWidth.valueProperty().addListener(e->{ //Wie die Spinner braucht das ein Listener. Wieder: mit Property
    arbeiten damti Listner auftaucht
112             canvas.setLineWidth(lineWidth.getValue()); //Slider haben einen Wert von 0 bis 100 und zwar als double
    Werte
113             canvas.drawFunctionCurve();
114 
115         });
116 
117 
118         canvas.drawFunctionCurve();
119         p.getChildren().addAll(s1,s2,s3,s4,btnReset,btnDraw,cbColor,lineWidth, canvas); //Am besten hier am Ende tun
    dann muss man keine Sorgen macahe dass man was
120                                                                                 //was dadrunter initialisiert. Reihefolge
    beeinflusst die Anordung in der GUI
121 
122         stage.setTitle(this.getClass().getSimpleName());
123         stage.setScene(scene);
124         stage.centerOnScreen();
125         stage.show();
126     }
127 }
128 
```