Datei: Moiree2.java

```
import java.awt.*;
import javax.swing.*;

public class Moiree2 extends JComponent{
  Color farbe;

  public Moiree2(Color c){
    farbe=c;
    setPreferredSize(new Dimension(150,150));
  }

  public void setFarbe(Color c){
    farbe=c;
  }

  public void paintComponent(Graphics g){
    g.setColor(farbe);
    int breit=getWidth();
    int tief=getHeight();
    for(int i=0; i<breit; i+=10){
      g.drawLine(0,     0,    i, tief);
      g.drawLine(breit, tief, i, 0   );
    }
    for(int i=0; i<tief; i+=10){
      g.drawLine(0,     0,    breit, i);
      g.drawLine(breit, tief, 0,     i);
    }
  }

  public static void main(String[] args){
    JFrame f=new JFrame("main von Moiree2.java");
    f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
    Container c=f.getContentPane();
    c.add(new Moiree2(Color.blue),BorderLayout.CENTER);
    f.pack();
    f.show();
  }
}
```

Wenn eine Instanz der vorliegenden Klasse sich etwas während der ganzen Lebenszeit merken muss, dann legt man für diese Eigenschaft eine Variable an, in Java Field genannt.
Die Farbe des Musters darf nicht vergessen werden, nachdem der Konstruktor abgearbeitet ist oder wenn der Zeichenvorgang in `paintComponent` abgeschlossen ist, deshalb gibt es das Field `farbe`.

Im Konstruktor wird festgelegt, was alles zu tun ist, wenn eine neue Instanz dieser Klasse erzeugt wird, also in diesem Fall mit `new Moiree2(eineFarbe)`.
Da hierbei auch ein ganzer Vorfahre erzeugt werden muss (hier ein `JComponent`), muss auch dessen Konstruktor aufgerufen werden. Das geht mit `super(...)`. Schreibt man das nicht ins Programm, so wird trotzdem automatisch ein solcher Aufruf eingefügt. Vor `farbe=c;` fügt der Compiler also noch den Befehl `super();` ein.

Da es von außen möglich sein sollte, die Farbe des Musters neu festzulegen, fügen wir die Methode `setFarbe(...)` ein. Ohne diese könnte man die Farbe von außen nur beim Erzeugungsvorgang festlegen, also im Konstuktor.

`paintComponent` wird von Windows aufgerufen, wenn die Instanz sich neu zeichnen muss, weil z.B. ein vorher darüber liegendes Fenster weggenommen wurde. Dafür liefert Windows ein Zeichenwerkzeug `Graphics` mit, das wir hier `g` genannt haben und natürlich unter diesem Namen ansprechen.

Mit dieser `main`-Methode wird die Klasse ein lauffähiges Programm. Es wird ein `JFrame` erzeugt und ein `Moiree2` hineingesetzt. Was ein `Moiree2` ist, wird (zufällig) in der gleichen Datei weiter oben festgelegt.

Wichtig: Die `main`-Methode könnte ebenso gut in einer anderen Datei alleine stehen, z.B. in `Starter.java`

Datei: Moiree2Frame.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;

public class Moiree2Frame extends JFrame implements ActionListener{
  Moiree2 moi;
  Random zufall;

  public Moiree2Frame(){
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    Container c=getContentPane();
    Moiree2 m=new Moiree2(Color.blue);
    c.add(m,BorderLayout.CENTER);
    moi=m;
    JButton b=new JButton("Farbe");
    c.add(b, BorderLayout.SOUTH);
    b.addActionListener(this);
    zufall=new Random();
  }

  public Moiree2Frame(String überschrift){
    this();//macht dasselbe wie Konstruktor mit ()
    setTitle(überschrift);
  }

  public void actionPerformed(ActionEvent e){
    moi.setFarbe(new Color(zufall.nextInt()));
    repaint();
  }

  public static void main(String[] args){
    Moiree2Frame f=new Moiree2Frame("main von Moiree2Frame.java");
    f.pack();
    f.show();
  }
}
```

Diesmal bauen wir den `JFrame` nicht in `main` zusammen, sondern legen im Konstruktor fest, dass er ein `Moiree2` beinhalten soll und einen `JButton`.
Übrigens sieht man hier, dass es auch mehrere Kunstruktoren geben kann für verschiedene Arten der Erzeugung:
Gibt man keine Überschrift mit an, wie in `new Moiree2Frame()`, wird der erste aufgerufen. Gibt man eine Überschrift mit,wie in new `Moiree2Frame("Hallo")`, wird der zweite aufgerufen.
Der zweite wiederum ist deshalb so kurz, weil er den ersten aufruft (mit `this();`) und nachträglich den Titel ändert (`setTitle` ist eine Methode, die der Vorfahre `JFrame` schon mitbringt).
`super()` wird vom Compiler wieder automatisch eingefügt.

Eine (Instanz von) `Moiree2Frame` erfüllt die Anforderungen eines `ActionListener`, weil sie hiermit die Methode `actionPerformed` bekommt. Bei wessen Actions sie benachrichtigt wird, wird nicht hier festgelegt, sondern beim Anmeldevorgang mit `...addActionListener(...)`.

`main` ist jetzt einfacher, weil mit der Erzeugung eines `Moiree2Frame` fast schon alles gemacht wird: `JButton` nach `SOUTH` legen, `Moiree2` nach `CENTER`, Anmelden als `ActionListener` beim `JButton`.