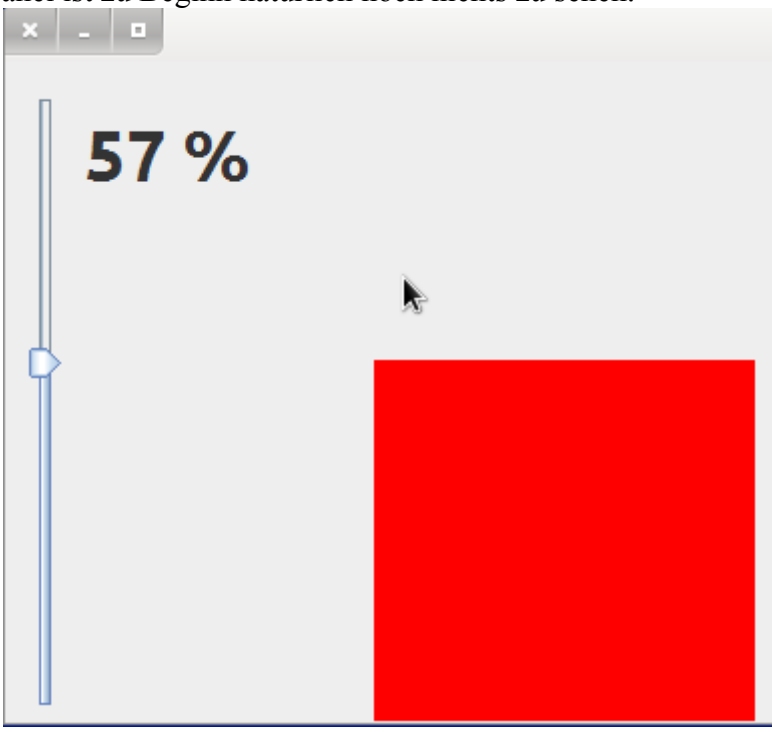
Einfache Grafiken erstellen mit Netbeans

Oft ist es sinnvoll, Sachverhalte mit Hilfe von Grafiken verständlicher zu machen. So kann eine Zahlenreihe mit Hilfe eines Diagramms schneller analysiert werden, als durch bloßes Betrachten der Zahlen selbst.
Oder ein Funktionsterm soll dargestellt werden oder ein Bild aus einer Datei.

In einem einfachen Beispiel soll ein Fenster mit einem Schieberegler (JSlider), einer Marke (JLabel) und einem Panel (JPanel) erstellt werden, in welchen durch Ändern der Position des Schiebereglers auch ein rotes Rechteck mit der entsprechenden Höhe dargestellt werden soll.

Fertige zunächst ein Formular mit Hilfe von Netbeans an, das diese drei Komponenten aufweist. In der Marke und im Panel ist zu Beginn natürlich noch nichts zu sehen.



Leider ist das Erstellen von Grafiken etwas komplizierter, als man es erwartet. Das liegt daran, dass Java selbst das Fenster immer wieder neu zeichnet. Dies ist zum Beispiel der Fall, wenn ein Teil des Fensters überdeckt war und nun, wieder im Vordergrund, neu gezeichnet werden muss.
Sollte auch die eigene Grafik überdeckt gewesen sein, so muss auch diese in genau dieser Situation neu dargestellt werden. Da man aber nicht genau weiß, wann das der Fall ist, muss man Java dazu bringen, bei jedem neuen Zeichnen auch die eigenen Grafiken auf den neuesten Stand zu bringen.

Jede grafische Komponente hat die Methode `paintComponent(Graphics g)`. Diese wird vom System immer aufgerufen, wenn ein Neuzeichnen ansteht. Deswegen sollte man auch die eigene Grafik in dieser Methode neu zeichnen lassen.

Nur wie kann man diese Methode `paintComponent` in Netbeans überschreiben?

Beim Anlegen eines Panels mit Hilfe von Netbeans kann man durch einen Rechtsklick auf dieses Panel im Kontextmenü den Punkt „Customize Code“ auswählen.
Der dann dargestellte Code entspricht dem, was Netbeans beim Anlegen des Panels aufruft. Wählt man nun aus dem Auswahlmenü „custom creation“ statt „default code“, so kann man diesen Bereich gemäß seinen Wünschen anpassen.

In unserem Fall soll das so aussehen:

```
jPanel1 = new javax.swing.JPanel(){
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        zeichneBild(g);
    }
};
```

super.paintComponenent(g) ruft die Methode von JPanel selbst auf, was immer zu empfehlen ist. Dann wird die Methode zeichneBild(g) aufgerufen, die sich dann um das Zeichnen kümmert.

Diese Methode können wir irgendwo im Quellcode unseres Fensters definieren:

```
public void zeichneBild (Graphics g){
        int b=jPanel1.getWidth();
        int h=jPanel1.getHeight();
        jLabel1.setText(wert+" %");
        int h2=wert*h/100;
        g.setColor(new Color(255,0,0));
        g.fillRect(0, h-h2, b, h2);
}
```

Sie holt sich Breite und Höhe des Panels, setzt den Text des Labels, ändert die Zeichenfarbe und zeichnet schließlich ein Rechteck.

Nun ist aber noch nicht klar, woher die Variable wert kommt, die in dem Programmausschnitt zu sehen ist.

Diese enthält den aktuellen Wert des JSliders. Um dies zu gewährleisten, müssen wir reagieren, sobald der Wert des Schiebereglers sich verändert. Dies gelingt durch Verwendung des Ereignisses stateChanged, welches die folgenden Zeilen aufrufen soll:

```
private void jSlider1StateChanged(javax.swing.event.ChangeEvent
evt) {
        wert=jSlider1.getValue();
        jPanel1.repaint();
}
```

Die Variable wert selbst muss in der obersten Zeile des Fensters definiert werden, so dass sie im ganzen Programm zur Verfügung steht.
jSlider1.getValue() liefert die aktuelle Position des Schiebereglers zurück und mit jPanel1.repaint() bittet man Java, das JPanel neu zu zeichnen, wodurch paintComponent automatisch aufgerufen wird und so auch unsere Grafik dargestellt wird.

Wichtiger Hinweis:
Die in diesem Kapitel skizzierte Vorgehensweise ist nur für Grafiken geeignet, die „schnell“ erzeugt werden können.
Will man hingegen kompliziertere Grafiken erstellen, deren Erzeugung jeweils 5 Sekunden dauert, so hat das vorgestellte Konzept den Nachteil, dass jedes Mal, wenn Java unser Fenster neu zeichnen muss (und das kommt öfter vor als man denkt), für 5 Sekunden nichts zu sehen ist.
In solchen Fällen rendert man die Grafik zunächst im Speicher (für den Benutzer unsichtbar) und tauscht das momentan sichtbare Bild durch das im Speicher erzeugte aus, sobald dieses fertig ist.

Hier nochmals die wesentlichen Punkte unseres Codes:

```
package panelgrafik;

import java.awt.Color;
import java.awt.Graphics;

public class Fenster extends javax.swing.JFrame {
    int wert=50; // hier wird die Variable wert definiert

    public Fenster() {
        initComponents();

    }

     [hier noch viele Zeilen Code von Netbeans]

    public void zeichneBild (Graphics g){
        int b=jPanel1.getWidth(); // breite holen
        int h=jPanel1.getHeight(); // höhe holen
        jLabel1.setText(wert+" %"); // prozentwert ausgeben
        int h2=wert*h/100;
        g.setColor(new Color(255,0,0)); // farbe auf rot setzen
        g.fillRect(0, h-h2, b, h2); // rechteck zeichnen

    }

    private void
jSlider1StateChanged(javax.swing.event.ChangeEvent evt) {
        wert=jSlider1.getValue(); // holt sliderwert
        jPanel1.repaint(); // bitte um neuzeichnen
    }

    [hier noch viele Zeilen Code von Netbeans]
}
```