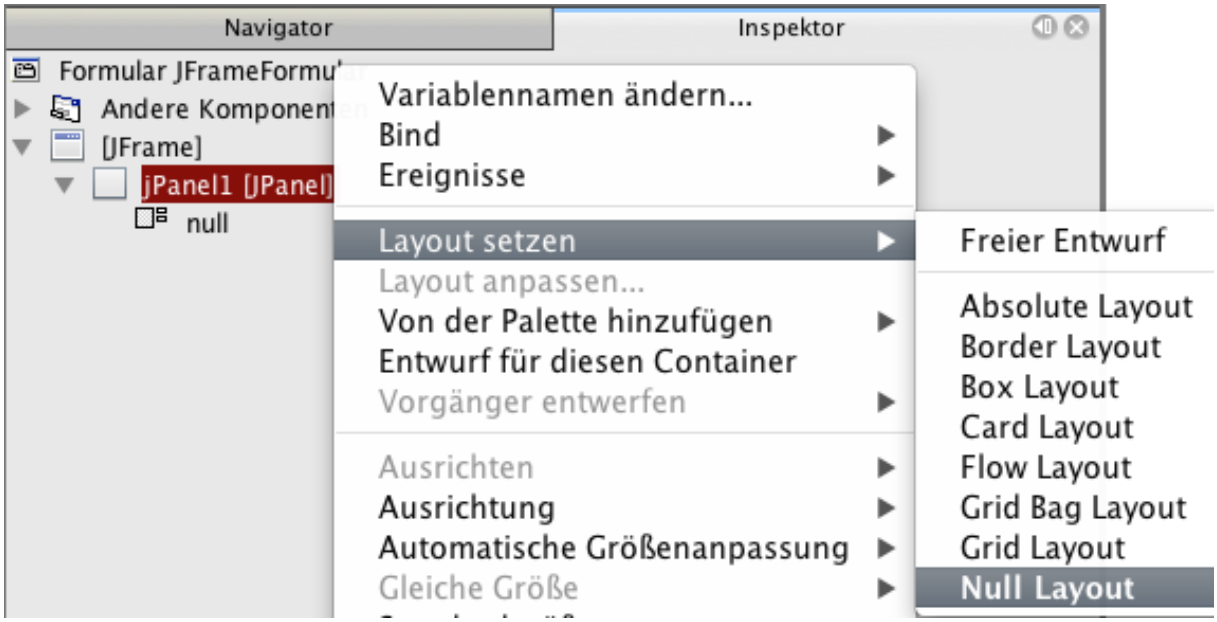
Tutorial: Bewegung von Objekten auf einer Swing-GUI durch Tastatureingabe

Grundeinstellungen

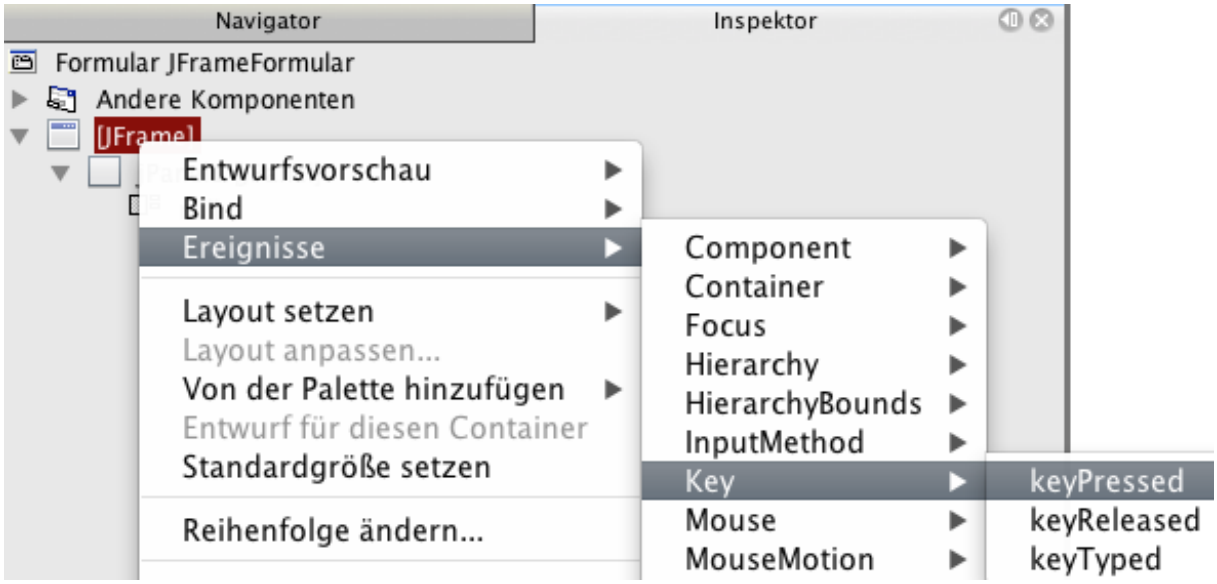
Layout des jPanels auf "null" setzen (sonst können Elemente nicht frei in ihrer Position verändert werden).



Tastenabfrage

Setzen Sie auf den JFrame einen Key Listener. Das ist eine Methode, die Tastatureingaben verarbeitet.

s.a. http://docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html

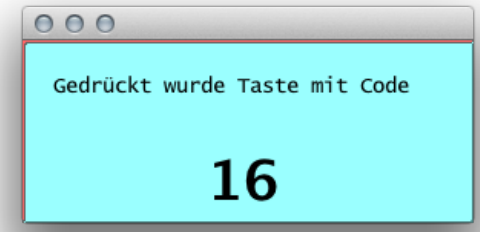


Der daraufhin generierten Methode formKeyPressed wird als Parameter ein Objekt ("evt") der Klasse KeyEvent übergeben. Dieses Objekt kennt Details über die erfolgte Tastatureingabe, z.B. den Code der betätigten Taste. Jede Taste hat einen eigenen Code, z.B.

Code	Taste
39	Pfeiltaste rechts
37	Pfeiltaste links
38	Pfeiltaste hoch
32	Space (Leertaste)
16	Shift (Umschalt)

Aufgabe

Holen Sie in der Methode formKeyPressed mit evt.getKeyCode() den Tastencode als int-Wert und geben Sie ihn auf einem Label aus.



Beachten Sie, dass Sie statt der Codes auch Konstanten verwenden können, z.B. VK_SPACE für die Leertaste oder VK_RIGHT für die Pfeiltaste rechts. Das ist meistens bequemer als mit Codes zu operieren.

Aufgabe

Programmieren Sie einige Reaktionen auf einige Tasten (z.B. Änderung der Hintergrundfarbe, Ausgabe eines Textes). Möglich wäre es z.B., immer ein Wort mit dem gedrückten Buchstaben auszugeben.
Das können Sie z.B. mit einer if-else-Verzweigung machen. Übersichtlicher ist aber switch.
Beachten Sie im folgenden Beispiel, dass VK_SPACE statisch ist, weswegen Sie VK_SPACE an KeyEvent schicken müssen (und nicht an evt).

```
private void formKeyPressed(java.awt.event.KeyEvent evt)
{
    switch(evt.getKeyCode())
    {
        case KeyEvent.VK_SPACE:
            jPanel1.setBackground(Color.pink);
            break;
    }
}
```

Komponenten bewegen

Weisen Sie z.B. einem Label eine neue Position zu mit der Methode setLocation(int x, int y).

Aufgabe
Sorgen Sie dafür, dass ein Label auf Tastendruck an eine andere Position springt.

Kennen sollten Sie die Ausmaße Ihres jPanels, um bspw. eine Bewegung verhindern zu können, bei der eine Komponente den sichtbaren Bereich verlässt. Schicken Sie hierzu dem jPanel bspw. die Methode getBounds.getMaxX().

Möglicherweise ist es sinnvoll, diese Werte in Konstanten zu speichern, bspw.

```
public class JFrameFormular extends javax.swing.JFrame
{
    private int LINKERRAND, RECHTERRAND, OBERERRAND, UNTERERRAND;

    /** Creates new form JFrameFormular */
    public JFrameFormular()
    {
        initComponents();
        this.LINKERRAND     = (int) jPanel1.getBounds().getMinX();
        this.RECHTERRAND    = (int) jPanel1.getBounds().getMaxX();
        this.OBERERRAND     = (int) jPanel1.getBounds().getMinY();
        this.UNTERERRAND    = (int) jPanel1.getBounds().getMaxY();
    }
```

Kennen sollten Sie auch die Eigenschaften der Komponente, die Sie manipulieren wollen. Wenn Sie vom Label "lblTest" die x-Koordinate brauchen, verwenden Sie

```
int posX = (int)lblTest.getLocation().getX();
```

Die Koordinaten beziehen sich auf die linke obere Ecke der Komponente.

Aufgabe
Ergänzen Sie die obere Aufgabe so, dass angezeigt wird, auf welcher Position sich das Label mit dem Tasten-Code jeweils befindet.

Damit wissen Sie alles, was Sie brauchen, um ein Label bspw. mit den Pfeiltasten zu steuern. Beachten Sie, dass Sie die Initialverzögerung, die bei dauerhaftem Drücken einer Taste auftritt, nicht über ihre Applikation eliminieren können.

Aufgabe
Versehen Sie ein (mehrere?) Label mit einem Bild (Eigenschaft "icon", Bild auswählen), am besten mit transparentem Hintergrund (animierte z.B. unter megagifs.de). Dieses Label stellt die Spielfigur dar.

Gestalten Sie den Spielhintergrund hübsch. Setzen Sie hinter Ihr jPanel ein Label, dem Sie über die Eigenschaft "icon" ein Hintergrundbild zuweisen. Damit das jPanel transparent wird, müssen Sie opaque auf "false" setzen. Platzieren Sie auch einige animierte Gifs auf dem Hintergrund.

Steuern Sie die Spielfigur mit den Pfeiltasten (Tipp: Eine Konstante SCHRITTWEITE definieren).

Seien Sie kreativ. Z.B.

- Die Spielfigur darf das Spielfeld nicht verlassen können.
- Wenn die Spielfigur das Spielfeld verlässt, geschieht irgendetwas.
- Die Bilder, die Sie auf der Oberfläche verteilt haben, können bspw. bei einem bestimmten Tastendruck (oder sonstigen Ereignis ... Kollision?) explodieren oder verschwinden oder ins Inventar aufgenommen werden.

Threads

Manchmal möchten Sie, dass verschiedene Vorgänge unabhängig voneinander parallel durchgeführt werden.

Beispiel 1: Wenn Sie die Taste "B" betätigen, soll der Hintergrund blinken (je nachdem ist das das Label für das Hintergrundbild oder das jPanel).

Richten Sie eine Klasse "Blinki" ein, die die Klasse Thread erweitert. Dem Konstruktor geben Sie als Parameter Ihr aktuelles JFrame-Objekt, damit Sie z.B. auf Methoden wie setBackground() zugreifen können.[1] Dann können Sie bspw. eine Schleife durchlaufen und dem Hintergrund in jedem Schleifendurchlauf eine neue, zufällige Zahl zuweisen.

Nach jedem Durchlauf stoppen Sie den Thread mit "sleep" für kurze Zeit, dass es nicht so flackert (Angabe der Zeit in Millisekunden). Sie müssen sleep mit einem try-catch umgeben.

```
public class Blinki extends Thread
{
    JFrameFormular j;

    public Blinki(JFrameFormular j)
    {
        this.j = j;
    }
    public void run()
    {
        for(int i=1;i<20;i++)
        {
            Random rand = new Random();
            float r = rand.nextFloat();
            float g = rand.nextFloat();
            float b = rand.nextFloat();
            Color c = new Color(r, g, b);
            j.setBackground(c);

            try {
                sleep(150);
            }
            catch (InterruptedException e)
            {

            }
        }
    }

}
```

Beispiel 2: Ihre Spielfigur soll auf Tastenklick eine Birne werfen. Allerdings muss sich während des Birnenflugs die Spielfigur weiter bewegen lassen.

Also richten Sie eine Klasse "Birnenwurf" ein, die die Klasse "Thread" erweitert. Diese Klasse bekommt im Konstruktor als Parameter die Objekte, auf die zugegriffen werden

---

[1] VIEL besser wäre es natürlich, Sie würden bspw. im JFrame eine Methode programmieren, die Zugriff auf den Label gibt und die Sie zum Zugriff benutzen würden. Mit der oben vorgeschlagenen Weise kriegen Sie z.B. Ärger, wenn Sie das Label durch eine andere Komponente ersetzen. Die Parameterübergabe ist aber übersichtlicher und einfacher.

soll, z.B. die Birne (als Wurfgeschoss) und die Spielfigur. Über die bekannten Methoden (spielfigur.getLocation().getX) können Sie bestimmen, wo sich die Spielfigur befindet. Mit einer Schleife können Sie die Birne von dieser Position aus durch das Feld wandern lassen (d.h.: abschießen).

Die Parameter sind vom Typ Component, z.B.:

```
public Birnenwurf(Component birne, Jlabel spielfigur)
```

Bevor Sie ein Label ("Birne") an den Konstruktor Birnenwurf übergeben, müssen Sie es in eine Component casten, z.B.

```
Component birne = (Component) lblBirne;
Birnenwurf birnenwurf = new Schuss(birne, lblSpielfigur);
```

In der (Thread-)Klasse, die für den Ablauf des Birnenwurfs verantwortlich ist, muss eine run()-Methode implementiert sein, dann dann beim Drücken bspw. der Space-Taste durch start() aufgerufen wird.

```
birnenwurf.start();
```