

Der MausBehälter

Der MausBehälter ergänzt den Behälter um Mausereignisse.

Wie beim Behälter werden Komponenten entweder beim Erzeugen im Behälter erzeugt oder später mit der Methode `hinzufuegen()` aus dem Ursprungsbehälter (meist dem Hauptfenster) hinzugefügt.

Signalisieren von Ereignissen:

Der Mausbehälter kann 8 verschiedene Ereignisse signalisieren.

Die beim Aufruf von `tuWas(int ID)` signalisierte ID ist die Summe aus der durch

`setzeLink(ITuWas linkObj, int BasisID)` übergebenen BasisID und der ID des Mausereignisses.

Wird das MausBehälterObjekt mit BasisID 10 initialisiert, so signalisiert der Mausbehälter das Ereignis RELEASE durch ID $(10+2) = 12$.

Die IDs der Mausereignisse:

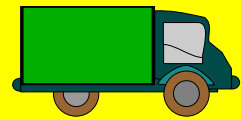
Ereignis	ID	Beschreibung
CLICK	0	Drücken und Loslassen
PRESS	1	Drücken der Maustaste
RELEASE	2	Loslassen der Maustaste
ENTER	3	Maus bewegt sich in den Behälter
EXIT	4	Maus verlässt den Behälter
DRAGGED	5	Bewegung mit gedrückter Maustaste
MOVED	6	Bewegung der Maus
WHEEL	7	Das Mousrad wurde bewegt

MausBehaelter

```

+ CLICK: int
+ PRESS: int
+ RELEASE: int
+ ENTER: int
+ EXIT: int
+ DRAGGED: int
+ MOVED: int
+ WHEEL: int

@ MausBehaelter()
@ MausBehaelter(int, int)
@ MausBehaelter(int, int, int, int)
@ MausBehaelter(IContainer)
@ MausBehaelter(IContainer, int, int, int, int)
+ getBasisComponente(): BasisComponente
+ hinzufuegen(IComponente): void
+ hinzufuegenUndAnpassen(IComponente): void
+ setzeZoomfaktor(double): void
+ setzeLink(ITuWas, int): void
+ ruecksetzenMaus(): void
+ setzeMausClick(): void
+ setzeMausPressRelease(): void
+ setzeMausEnterExit(): void
+ setzeMausDraggedMoved(): void
+ setzeMausRad(): void
+ setzeMausereignisse(int): void
+ setzeAlleMausereignisse(): void
+ mausAktion(): boolean
+ getMX(): int
+ getMY(): int
+ getClickCount(): int
+ getButton(): int
+ getShift(): boolean
+ getCtrl(): boolean
+ getAlt(): boolean
+ getRotation(): int
+ getXPos(): int
+ getYPos(): int
+ setzeGroesse(int, int): void
+ setzePosition(int, int): void
+ setzeDimensionen(int, int, int, int): void
+ verschieben(int, int): void
+ sichtbarMachen(): void
+ unsichtbarMachen(): void
+ setzeMitRaster(boolean): void
+ setzeDeltaX(int): void
+ setzeDeltaY(int): void
+ add(Component, int): Component
+ setzeKomponentenKoordinaten(JComponent, int, int, int, int): void
+ setzeKomponentenGroesse(JComponent, int, int): void
+ setzeKomponentenPosition(JComponent, int, int): void
+ validate(): void
+ getPanel(): JPanel
+ getBehaelterZoom(): double
    
```



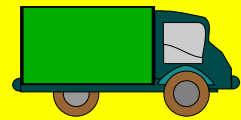
Einschalten der Mausereignisse

Die Maus signalisiert bis zu 8 verschiedene Ereignisse. Oft braucht man nur einzelne. Daher muss vor der Verwendung der Ereignisse das gewünschte Ereignis eingeschaltet werden. Folgende Methoden stellen Gruppen von Ereignissen ein. Gruppen werden zusätzlich zu bereits eingestellten gesetzt.

ruecksetzenMaus()	Alle Mausereignisse deaktivieren
setzeMausClick()	Click-Ereignis
setzeMausPressRelease()	Drücken und Loslassen
setzeMausEnterExit()	Mit der Maus die Komponente betreten bzw. verlassen
setzeMausDraggedMoved()	Mit gedrückter Maustaste (dragged) und ohne gedrückte Maustaste (moved) bewegen. Die Ereignisse finden bei gedrückter Maustaste bis zum Loslassen der Maustaste statt, also auch außerhalb des Mausbehälters!
setzeMausRad	Aktiviere das Mousrad
setzeAlleMausereignisse	Alle Ereignisse setzen.
setzeMausereignisse (int ereignisse)	Beim Parameter ereignisse wird pro Ereignis ein Bit gesetzt. Folgende Aufzählung aktiviert all. Für die eigene Zusammenstellung die gewünschten Zeilen übernehmen: (1 << MausBehaelter.CLICK) (1 << MausBehaelter.PRESS) (1 << MausBehaelter.RELEASE) (1 << MausBehaelter.ENTER) (1 << MausBehaelter.EXIT) (1 << MausBehaelter.DRAGGED) (1 << MausBehaelter.MOVED) (1 << MausBehaelter.WHEEL)

Zu Beachten:

Einige Mausereignisse führen zu mehreren Ereignissen:
Ein Click-Ereignis geht immer ein Press- und ein Release-Ereignis voraus.
Einem Doppelclick geht eine EinfachClick voraus usw.

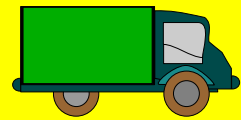


Status beim Eintreten des Mausereignisses:

Methode mit Rückgabetyt	Beschreibung
<code>public boolean</code> <code>mausAktion()</code>	Zeigt, dass eine Mausektion stattgefunden hat. Diese Methode kann verwendet werden, wenn ohne Callback der mausstatus abgefragt werden soll.
<code>public int</code> <code>getMX()</code>	x-Koordiante der Maus relativ zum Behaelter
<code>public int</code> <code>getMY()</code>	y-Koordiante der Maus relativ zum Behaelter
<code>public int</code> <code>getClickCount()</code>	Anzahl der Clicks (für Doppelclick usw.)
<code>public int</code> <code>getButton()</code>	Auslösende Maustaste: Links = 0 ...
<code>public boolean</code> <code>getShift()</code>	War die Umschalttaste gedrückt?
<code>public boolean</code> <code>getCtrl()</code>	War die Steuerungstaste gedrückt?
<code>public boolean</code> <code>getAlt()</code>	War die Altaste gedrückt?
<code>public int</code> <code>getRotation()</code>	Anzahl der Rotations-Ticks (rückwärts negativ)
<code>public int</code> <code>getXPos()</code>	x-Position des Mausbehälters
<code>public int</code> <code>getYPos()</code>	y-Position des Mausbehälters

Die Angabe sind die Angaben beim Eintritt des letzten Mausereignisses.

(Man sollte daher nicht benötigte Ereignisse deaktivieren. Sie verfälschen eventuell benötigte (Positions-)Daten.)



Einsatz des MausBehälters am Beispiel B_Maustest

```
public class B_Maustest implements ITuWas {
```

```
    MausBehaelter    obj;
```

```
    Kreis            aktion;  
    Taktgeber        takt;  
    Rechteck         innen;
```

```
    Ausgabe          id;  
    Ausgabe          x;  
    Ausgabe          y;  
    Ausgabe          clicks;  
    Ausgabe          taste;
```

```
    Ausgabe          shift;  
    Ausgabe          ctrl;  
    Ausgabe          alt;  
    Ausgabe          rotation;
```

```
public B_Maustest() {  
    aktion = new Kreis();  
    aktion.setzeFarbe("rot");
```

```
    takt = new Taktgeber(this, 99);
```

```
    obj = new MausBehaelter(100, 200, 300, 300)  
    obj.setzeMausereignisse((1 << MausBehaelter.CLICK)  
        | (1 << MausBehaelter.PRESS) | (1 << MausBehaelter.RELEASE)  
        | (1 << MausBehaelter.ENTER) | (1 << MausBehaelter.EXIT)  
        | (1 << MausBehaelter.DRAGGED) | (1 << MausBehaelter.MOVED)  
        | (1 << MausBehaelter.WHEEL));
```

```
    innen = new Rechteck(obj, 0, 0, 300, 300);
```

```
    id = new Ausgabe("id", 100, 0, 100, 50);  
    x = new Ausgabe("x", 200, 0, 100, 50);  
    y = new Ausgabe("y", 300, 0, 100, 50);
```

```
    clicks = new Ausgabe("Clicks", 400, 0, 100, 50);
```

```
    taste = new Ausgabe("Taste", 100, 100, 100, 50);
```

```
    shift = new Ausgabe("Shift", 200, 100, 100, 50);
```

```
    ctrl = new Ausgabe("CTRL", 300, 100, 100, 50);
```

```
    alt = new Ausgabe("ALT", 400, 100, 100, 50);
```

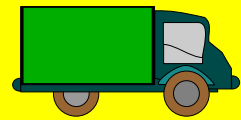
```
    rotation = new Ausgabe("Rot", 50, 200, 100, 50);
```

```
    obj.setzeLink(this, 0);  
} // Ende Konstruktor
```

Der Takt setzt den Kreis 100 ms
nach dem Mausereignis zurück

Der Mausbehälter
Es werden alle Ereignisse aktiviert

Ausgabeelemente



```
public void anzeigen(int ID) {
    id.setzeAusgabertext("ID " + ID);
    x.setzeAusgabertext("X:" + obj.getMX());
    y.setzeAusgabertext("Y:" + obj.getMY());
    clicks.setzeAusgabertext("CLC:" + obj.getClickCount());
    taste.setzeAusgabertext("T:" + obj.getButton());

    if (ID == 1){
        innen.setzeFarbe("blau");
    }
    if (ID == 2){
        innen.setzeFarbe("weiss");
    }

    if (obj.getShift())
        shift.setzeFarbe("rot");
    else
        shift.setzeFarbe("gelb");

    if (obj.getCtrl())
        ctrl.setzeFarbe("rot");
    else
        ctrl.setzeFarbe("gelb");

    if (obj.getAlt())
        alt.setzeFarbe("rot");
    else
        alt.setzeFarbe("gelb");
    rotation.setzeAusgabertext("R:" + obj.getRotation());
}
// Anzahl der Ticks des Mousrads
public int getRotation() {
    return obj.getRotation();
}
```

Die Anzeigemethode
Je nach ID wird die Anzeige gesetzt

```
public void tuWas(int ID) {
    switch (ID) {
        case 99:
            aktion.fuellen();
            break;
        default:
            takt.einmal(100);
            aktion.rand();
            anzeigen(ID);
    }
}
```

Die Methode tuWas wird von
der Mauskomponente und
dem Taktgeber aufgerufen.

Der Taktgeber(ID 99) füllt den Kreis (links oben)

Die Maus-Signale (sonstige Ids = default)
Starten takt. takt signalisiert einmalig in 100 ms
Der Kreis wird zum Rand
Dann wird die Methode anzeigen aufgerufen

```
public static void main(String[] args) {
    new B_Maustest();
}
```

Programmstart