

Übersicht:

Erstellung eigener Dialoge.....	1
Der modale Dialog.....	1
Der nicht modale Dialog.....	1
Die Dialogklasse der JGUIToolbox.....	2
Programmfragment:.....	2
Für den modalen Dialog:.....	2
Beispiel einer Klasse, die ein modales Dialogelement verwendet.....	3
Beispiel einer Klasse, die ein nicht modales Dialogelement verwendet.....	4
Der Dialog als eigenständige Klasse.....	5
Die Dialog-Klasse:.....	5
Die Klasse, die den Dialog verwendet:.....	6
Kommunikation zwischen den Fenstern.....	7
Hauptfenster → DialogObjekt.....	7
Dialogobjekt → Hauptfenster.....	7
Warnung:.....	7
Die universelle Dialog-Klasse:.....	7

Erstellung eigener Dialoge

Ein Dialogfenster ist ein Fenster, das über dem Hauptfenster erscheint.

Es gibt zwei Arten von Dialogen:

Der modale Dialog:

Die Eingabe des **Hauptfenster** ist **gesperrt**, bis das Dialogfenster geschlossen wird.

Der nicht modale Dialog:

Ein Fenster, das über dem Hauptprogramm schwebt.

Tastatureingaben gehen an das Fenster, das den Fokus hat.

Der modale Dialog

Die Eingabe des Hauptfensters ist **während der Sichtbarkeit** des modalen Dialogs blockiert. Der Status des Dialogfelds kann vom Hauptfenster aus erst nach dem Schließen des Dialogfensters des Dialogs abgefragt werden. Der Dialog ist ein Objekt, das nach Beenden des Dialogs existiert. Die Elemente des Dialogs können abgefragt werden.

Der nicht modale Dialog

Die Eingabe ist beim **Dialogfenster** und beim **Hauptfenster** möglich.

Startet man den **nicht modalen Dialog**, so wird das Fenster des nicht modalen Dialogs sichtbar. Dann läuft das Hauptprogramm weiter. Damit beide Fenster aufeinander wirken können, muss eine Kommunikationsmöglichkeit zwischen beiden installiert sein.

Die Dialogklasse der JGUIToolbox

Die Klasse kapselt die JDialog-Klasse von Java.

Der Konstruktor ist **D_JGUIDialog(String titel, boolean modal)**

Parameter:

Titel: In der Fensterleiste angezeigter Text.

Modal: `MODAL = true ;`
`NICHTMODAL = false ;`

Die Klasse stellt einen **Behälter** zur Verfügung.

Die Methode **leseContainer()** findet den Behälter.

Die Komponenten der Toolbox können damit im Behälter des Dialogs erzeugt werden.

Programmfragment:

```
// Erzeugen eines modalen Dialogs
dialog = new D_JGUIDialog("Einstellungen", true );
// behälter enthält den behälter des Dialogs
behaelter = dialog.leseContainer();
// Ein Eingabefeld und eine Taste werden erzeugt.
eingabe = new Eingabefeld(behaelter, "...", 0, 0, 400, 80);
ja = new Taste(behaelter, "Übernehmen", 50, 100, 150, 50);
nein = new Taste(behaelter, "Abbruch", 200, 100, 150, 50);
```

D_JGUIDialog
☒ MODAL: boolean
☒ NICHTMODAL: boolean
☒ D_JGUIDialog(String, boolean)
☒ setzeSichtbar(boolean): void
☒ setSize(int, int): void
☒ leseContainer(): IContainer
☒ setzeID(int): void
☒ setzeLink(ITuWas): void
☒ setzeLink(ITuWas, int): void
☒ tuWas(int): void
☒ leseCloseID(): int
☒ resetCloseID(): void

Für den modalen Dialog:

Der modale Dialog braucht Elemente, die den Dialog schließen.

Dazu muss diesem Element ein Link auf das Dialogelement übergeben werden.

Nach dem Schließen des Dialogs kann durch die Methode **int leseCloseID()** die ID des schließenden Elements abgefragt werden.

Bei mehreren schließenden Elementen brauchen diese Elemente jeweils eigene IDs.

Fortsetzung des Programmfragments:

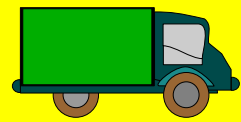
```
ja.setzeID(1);
ja.setzeLink(dialog);
nein.setzeID(0);
nein.setzeLink(dialog);
```

Der Dialog wird jetzt durch Klicken auf die Übernehmen- und die Abbruch-Taste beendet.

Nach beenden des Dialogs durch Übernehmen erhält man durch leseCloseId() den Wert 1.

Setzt das Hauptprogramm einen Link auf den **Dialog** mit setzeLink, so wird beim Beenden des Dialogs die Methode tuWas(int ID) des übergebenen Objekts aufgerufen. Die übergebene ID ist die Summe aus der ID des Dialogs und des Dialogobjekts, das den Dialog beendet hat.

Die Methode **setzeSichtbar(true)** startet den Dialog.



Beispiel einer Klasse, die ein **modales** Dialogelement verwendet.

```
public class Dialog_Modal implements ITuWas{
    // Dialog
    D_JGUIDialog dialog;
    // Container der Dialog-Komponente
    IContainer behaelter;
    Eingabefeld eingabe;
    Taste ende;
    Taste abbruch;
    // Das Hauptfenster
    Taste dialogstart;
    Eingabefeld hauptfensterMeldung;

    public Dialog_Modal() {
        // Dialog anlegen
        dialog = new D_JGUIDialog("Einstellungen", D_JGUIDialog.MODAL);
        dialog.setzeID(1000); // 1000 = BasisID Dialog
        // Größe des Dialogfensters einstellen
        dialog.setSize(400, 300);
        // Behälter für Dialogkomponenten lesen
        behaelter = dialog leseContainer();

        // Eingabefeld im Dialog erzeugen
        eingabe = new Eingabefeld(behaelter, "Text eingeben", 0, 0, 400, 80);
        eingabe.setzeID(100); // 100 = ID Eingabefenster Dialog
        eingabe.setzeLink(this); // Meldung an Hauptfenster
        // Beenden-Taste in den Dialog-Behälter einfügen
        ende = new Taste(behaelter, "Beenden", 50, 100, 150, 50);
        ende.setzeID(101); // 101 = ID Beenden-Taste
        ende.setzeLink(dialog); // Meldung an Dialog
        // Beenden des Dialogs! // Beenden-Taste beendet den Dialog
        // Abbruch-Taste in den Dialog-Behälter einfügen
        abbruch = new Taste(behaelter, "Abbruch", 50, 200, 150, 50);
        abbruch.setzeID(102); // 102 = ID Abbruchtaste
        abbruch.setzeLink(dialog); // Meldung an den Dialog
        // Beenden des Dialogs! // Abbruch-Taste beendet den Dialog

        // Hauptfenster
        dialogstart = new Taste("Starte Dialog", 50, 300, 200, 80);
        dialogstart.setzeID(1); // 1 = ID Dialogstart
        dialogstart.setzeLink(this); // Startet den Dialog
        // Eingabefenster im Hauptfenster
        hauptfensterMeldung = new Eingabefeld("Eingabe", 50, 400, 300, 80);
    }

    public void tuWas(int ID) {
        if ( ID == 1){ // Dialog wird gestartet
            dialog.setzeSichtbar(true);
            // Ausführung wird angehalten bis Dialog beendet
            if(dialog leseCloseID()== 101){ // Beenden gedrückt
                hauptfensterMeldung.setzeAusgabetext(eingabe leseText());
            }
        } else if ( ID == 100 ){ // Enter im Eingabefeld des Dialogs
            // Der Text wird in Großbuchstaben verwandelt
            eingabe.setzeAusgabetext(eingabe leseText().toUpperCase());
        }
    }

    // Main-Methode. Kann in BlueJ entfallen
    public static void main(String [] args ) {
        new Dialog_Modal();
    }
}
```

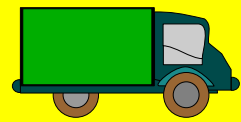
Die Dialogkomponente

Elemente auf dem Dialogelement

Hauptfenster

Beenden des Dialogs!

Beenden des Dialogs!



Beispiel einer Klasse, die ein **nicht modales Dialogelement** verwendet.

```
public class Dialog_NichtModal implements ITuWas{
    // Dialog
    D_JGUIDialog dialog;
    // Container der Dialog-Komponente
    IContainer behaelter;
    Eingabefeld eingabe;
    Taste uebernehmen;

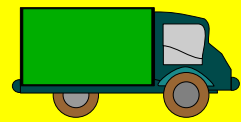
    // Das Hauptfenster
    Taste dialogstart;
    Eingabefeld hauptfensterMeldung;

    public Dialog_NichtModal() {
        // Dialog anlegen
        dialog = new D_JGUIDialog("Einstellungen", D_JGUIDialog.NICHTMODAL);
        dialog.setzeID(1000); // 1000 = BasisID Dialog
        dialog.setzeLink(this); // Meldung des Dialogs an das Hauptfenster
        // Größe des Dialogfensters einstellen
        dialog.setSize(400, 300);
        // Behälter für Dialogkomponenten lesen
        behaelter = dialog.leseContainer();
        // Eingabefeld im Dialog erzeugen
        eingabe = new Eingabefeld(behaelter, "Text eingeben", 0, 0, 400, 80);
        eingabe.setzeID(100); // 100 = ID Eingabefenster Dialog
        eingabe.setzeLink(this); // Meldung an Hauptfenster
        // Beenden-Taste in den Dialog-Behälter einfügen
        uebernehmen = new Taste(behaelter, "Übernehmen", 50, 100, 150, 50);
        uebernehmen.setzeID(101); // 101 = ID Übernehmen-Taste
        uebernehmen.setzeLink(dialog); // Meldung an Dialog
        // Beenden des Dialogs!

        // Hauptfenster
        dialogstart = new Taste("Öffne Dialog", 50, 300, 200, 80);
        dialogstart.setzeID(1); // 1 = ID Dialogstart
        dialogstart.setzeLink(this); // Startet den Dialog
        // Eingabefenster im Hauptfenster
        hauptfensterMeldung = new Eingabefeld("Eingabe", 50, 400, 300, 80);
    }

    public void tuWas(int ID) {
        if ( ID == 1){ // Dialog wird gestartet
            // Dialogeingabe wird auf Hauptfenstereingabe gesetzt
            eingabe.setzeAusgabertext(hauptfensterMeldung.leseText());
            dialog.setzeSichtbar(true); // Das Programm läuft sofort weiter
        } else if ( ID == 100 ){ // Enter im Eingabefeld des Dialogs
            // Der Text wird in Großbuchstaben verwandelt.
            // Text wird auch im Hauptfenster gesetzt
            eingabe.setzeAusgabertext(eingabe.leseText().toUpperCase());
            hauptfensterMeldung.setzeAusgabertext(eingabe.leseText());
        } else if (ID == (1000 + 101)){ // Der Dialog meldet weiter! ID = DialogID + ID des Elements
            hauptfensterMeldung.setzeAusgabertext(eingabe.leseText());
        }
    }

    // Main-Methode. Kann in BlueJ entfallen
    public static void main(String [] args ) {
        new Dialog_NichtModal();
    }
}
```



Der Dialog als eigenständige Klasse

In den beiden Beispielen wird die Dialogkomponente als Teil der Hauptklasse eingesetzt. Übersichtlicher ist es, für den Dialog eine eigene Klasse zu erstellen. Der nicht modalen Dialog wird zerlegt in eine Dialog-Klasse und das Hauptfenster in der dann die Dialogkomponente verwendet wird. Das wird hier gezeigt am Beispiel des Beispiels des nicht modalen Dialogs.

Die Dialog-Klasse:

```
public class DialogNichtmodal implements ITuWas {
    // Dialog
    D_JGUIDialog dialog;
    // Container der Dialog-Komponente
    IContainer behaelter;
    Eingabefeld eingabe;
    Taste uebernehmen;

    // Verbindung zum Hauptfenster
    DialogHauptklasse hauptfenster ;

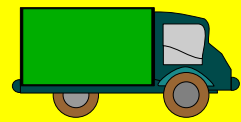
    public DialogNichtmodal() {
        // Dialog anlegen
        dialog = new D_JGUIDialog("Einstellungen", D_JGUIDialog.NICHTMODAL);
        dialog.setzeID(1000); // 1000 = BasisID Dialog
        dialog.setzeLink(this); // Meldung des Dialogs an das Hauptfenster
        // Größe des Dialogfensters einstellen
        dialog.setSize(400, 300);
        // Behälter für Dialogkomponenten lesen
        behaelter = dialog leseContainer();
        // Eingabefeld im Dialog erzeugen
        eingabe = new Eingabefeld(behaelter, "Text eingeben", 0, 0, 400, 80);
        eingabe.setzeID(100); // 100 = ID Eingabefenster Dialog
        eingabe.setzeLink(this); // Meldung an Hauptfenster
        // Beenden-Taste in den Dialog-Behälter einfügen
        uebernehmen = new Taste(behaelter, "Übernehmen", 50, 100, 150, 50);
        uebernehmen.setzeID(101); // 101 = ID Übernehmen-Taste
        uebernehmen.setzeLink(dialog); // Meldung an Dialog
    }

    public void tuWas(int ID) {
        if (ID == 100) { // Enter im Eingabefeld des Dialogs
            // Der Text wird in Großbuchstaben verwandelt.
            // Text wird auch im Hauptfenster gesetzt
            eingabe.setzeAusgabertext(eingabe leseText().toUpperCase());
            hauptfenster.hauptfensterMeldung.
                setzeAusgabertext(eingabe leseText());
        } else
            // Die Taste Übernehmen wurde gedrückt
            if (ID == (1000 + 101)) {
                hauptfenster.hauptfensterMeldung.
                    setzeAusgabertext(eingabe leseText());
            }
    }
}
```

Die Klasse DialogNichtmodal hat eine Referenz auf das Hauptfenster. Sie kann daher auf Komponenten des Hauptfensters zugreifen.

Warnung: Es muss sichergestellt sein, dass diese Referenz nicht ins Nichts zeigt!

Sichere Variante: **if(hauptfenster != null) hauptfenster. ...** (Auch für Unter-Elemente!)



Die Klasse, die den Dialog verwendet:

```
public class DialogHauptklasse implements ITuWas{
    // Das Dialog-Objekt
    DialogNichtmodal dialogfenster ;
    // Das Hauptfenster
    Taste dialogstart;
    Eingabefeld hauptfensterMeldung;

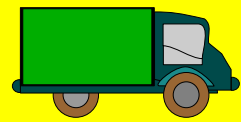
    public DialogHauptklasse() {
        // Die Taste zum Starten des Dialogs
        dialogstart = new Taste("Öffne Dialog",50,300,200,80);
        dialogstart.setzeID(1); // 1 = ID Dialogstart
        dialogstart.setzeLink(this); // Startet den Dialog
        // Eingabefenster im Hauptfenster
        hauptfensterMeldung = new Eingabefeld("Eingabe", 50,400,300,80);

        // Dialog anlegen
        dialogfenster = new B_MDialogDialogNichtmodal();
        // Die Referenz auf das Huptfenster wird beim Dialog gesetzt
        dialogfenster.hauptfenster = this ;
    }

    public void tuWas(int ID) {
        if ( ID == 1){ // Dialog starten
            // Dialog-Eingabefeld wird mit dem Inhalt
            //des Hauptfenster-Eingabefelds überschrieben
            dialogfenster.eingabe.
                setzeAusgabebetext(hauptfensterMeldung leseText());
            dialogfenster.dialog.setzeSichtbar(true);
            // Das Programm läuft sofort weiter
        } else if (ID == (1000 + 101)){
            // Signal vom Dialog
            hauptfensterMeldung.
                setzeAusgabebetext(dialogfenster.eingabe leseText());
        }
    }

    // Main-Methode. Kann in BlueJ entfallen
    public static void main(String [] args ) {
        new B_MDialogHauptklasse();
    }
}
```

Da der Dialog auf Elemente des Hauptfensters zugreifen kann, sollte der Dialog erst dann zu erzeugt werde, wenn alle Komponenten des Hauptfensters angelegt sind. Daher steht der Aufruf des Konstruktors der Dialogklasse am Ende des Konstruktors der Hauptklasse.



Kommunikation zwischen den Fenstern

Hauptfenster → DialogObjekt.

Das Hauptfenster hat aus der Erzeugung des Dialog-Objekts eine Referenz auf das Dialogobjekt. Darüber können Methoden des Dialogobjekts aufgerufen werden und Attribute des Dialogobjekts gelesen/verändert werden.

Dialogobjekt → Hauptfenster

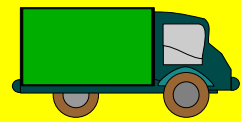
Dem Dialogobjekt wird das Hauptfenster bekannt gemacht. Das kann z.B. eine Referenz auf das Hauptfenster sein. Dadurch kann dann auch das Dialogobjekt Methoden des Hauptfensters verwenden.

Auch beim modalen Dialog möglich! Der modale Dialog blockiert die Eingabe des Hauptfensters, nicht die Methoden des Hauptprogramms!

Warnung:

Vor dem ersten Zugriff auf ein Element des anderen Objekts muss sichergestellt sein, dass dieses existiert. Greift man auf eine Objekt in einem anderem Objekt zu, so gilt das Gleiche! Für Sicherheitsbewusste: **if(referenz != null) referenz., auch für Unter-Elemente.**

Da der Dialog auf Elemente des Hauptfensters zugreifen kann, ist es sicherer, den Dialog erst dann zu erstellen, wenn alle Komponenten des Hauptfensters angelegt sind.



Die universelle Dialog-Klasse:

Möchte man eine universell einsetzbare Dialog-Klasse erstellen, so muss man für die Kommunikation Dialogobjekt → Hauptfenster eine **universelle Form** einsetzen, da hier die Dialogklasse **sein Hauptfenster nicht kennen kann**.

Es bietet sich die in der Toolbox verwendete Art der Kommunikation an:

Die Dialogklasse enthält das folgende Code-Fragment:

```
int basisID = 0; // Basis-ID der Dialog-Klasse
/** Setzen der BasisID des Dialogs */
public void setzeID(int ID) {
    basisID = ID;
}
ITuWas linkObj; // Referenz auf die zu benachrichtigende Komponente
/** Setzen des Links auf die zu benachrichtigende Komponente */
public void setzeLink(ITuWas linkObj) {
    obj.setzeLink(linkObj);
}
/** aus historischen Gründen beides zusammen */
public void setzeLink(ITuWas linkObj, int ID) {
    this.linkObj = linkObj;
    basisID = ID;
}
```

Die Dialog-Komponente enthält bereits die Methode tuWas(int ID), die von den aktiven Komponenten des Dialogs aufgerufen wird. Bei IDs, die Nachrichten an andere Objekte weiterleiten sollen wird dann für den internen Code der Code eingefügt:

```
if (linkObj != null)
    linkObj.tuWas(basisID + ID);
```

Über die ID kann dann die tuWas-Methode des aufrufenden Fensters den Dialog als Quelle des Signals identifizieren.