

Der Taktgeber

Der Taktgeber kann bei graphischen Aktionen als Ersatz für Schleifen verwendet werden. Der Taktgeber ruft in der tuWas-Methode wiederholt die gewünschte Aktion auf.

Sollen mehrere Animationsstränge parallel laufen, bereitet die Wiederholung Probleme. Alles muss in einer Schleife integriert sei, da diese Schleife alle anderen anhält.

Jede Komponente kann einen eigenen Taktgeber installieren. Es ist keine zentrale Logik nötig. Die Methode tuWas zeichnet. Sie unterbricht das System nur kurz.

Der Methode **mehrfach** wird als Parameter die gewünschte Anzahl übergeben. Die Methode **endlos** ersetzt die Endlos-Schleife.

Die Methode **einmal** kann als definierte Zeitverzögerung benutzt werden.

Taktgeber	
wurdeSignalisiert	Seit der letzten Abfrage hat der Timer signalisiert
warteBisTaktsignal	Pause bis Timer signalisiert
setzeZeitZwischenAktionen	Zeit zwischen zwei Timersignalen
setzeAnfangszeitverzoeigerung	Zeit von Start() bis zum ersten Timersignal
mehrfach(int anzahl)	anzahl Timersignale
einmal() und einmal(int delay)	Ein einziges Timersignal (delay in ms)
endlos()	immer
stop()	Stop des Timers
boolean laufend()	Der Timer ist noch aktiv

Die Methoden mehrfach, einmal, endlos starten den Timer.

Taktgeber
<ul style="list-style-type: none"> [-] t: Timer [-] delay: int [-] startDelay: int [-] timerSignal: boolean [-] anzahl: int [-] begrenzteAnzahl: boolean [-] linkObj: ITuWas [-] id: int
<ul style="list-style-type: none"> ⊙ Taktgeber() ⊙ Taktgeber(ITuWas, int) ⊖ tuWas(): void ⊕ setzeLink(ITuWas, int): void ⊕ wurdeSignalisiert(): boolean ⊕ warteBisTaktsignal(): void ⊕ setzeZeitZwischenAktionen(int): void ⊕ setzeAnfangszeitverzoeigerung(int): void ⊕ mehrfach(int): void ⊕ einmal(): void ⊕ einmal(int): void ⊕ endlos(): void ⊕ stop(): void ⊕ laufend(): boolean

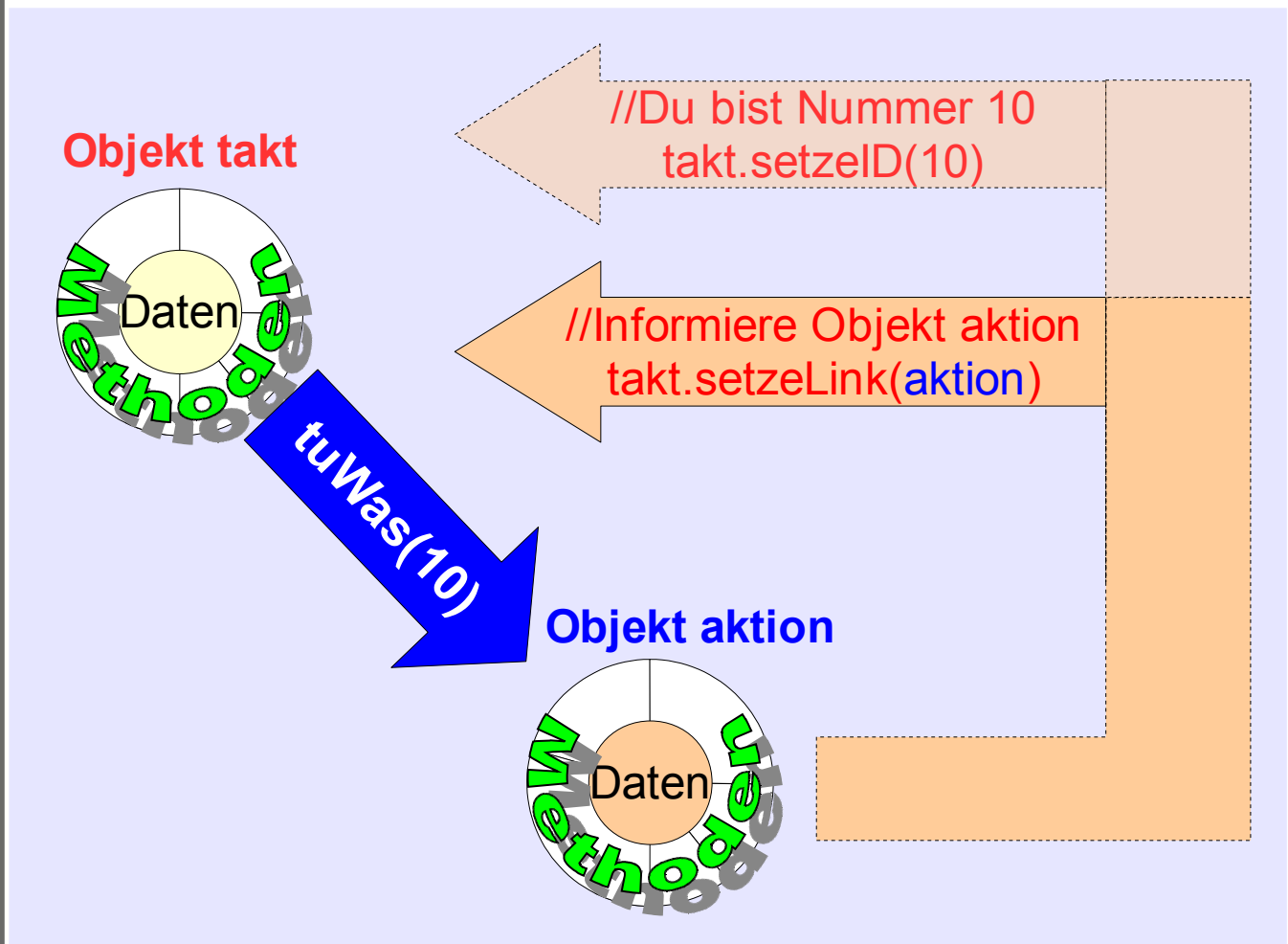
Kommunikation

Objekte kommunizieren in Java über **Methoden**.

Das Objekt **takt** der Klasse **Taktgeber** signalisiert dem Objekt **aktion**, indem es eine Methode **tuWas** von **aktion** aufruft.

Ablauf:

(wie auch bei den übrigen aktiven Komponenten der JGUIToolbox):



[1] Dem Objekt **takt** wird die **ID 10** zugewiesen: **setzeID(10)**

(Kann entfallen, wenn die Identität des signalisierenden Objekts nicht wichtig ist)

Das Objekt **takt** wird informiert, dass es das Objekt **aktion** informieren soll

setzeLink(aktion)

[2] Das Objekt **takt** informiert das Objekt **aktion** durch Aufruf der Methode **tuWas(10)**.

Das Objekt **takt** muss sicher sein, dass das Objekt **aktion** die Methode **tuWas** besitzt.

Deshalb **implements ITuWAS** im Klassenkopf der dazugehörenden

Klasse **Bewegung**.

Einsatz des Taktgebers am Beispiel:

```
/**
```

```
 * ein Kreis bewegt sich.
```

```
 * Aktion durch Taktgeber
```

```
 */
```

```
public class Bewegung implements ITuWas {
```

```
    // Variablen und Objekte
```

```
    private Kreis lampe ;
```

```
    private Taktgeber takt ;
```

Deklaration des Objekts takt

```
    /**
```

```
     * Konstruktor für Objekte der Klasse Bewegung
```

```
    */
```

```
    public Bewegung()
```

```
    {
```

```
        // Objekte initialisieren
```

```
        lampe = new Kreis(100, 100,50);
```

```
        lampe.setzeFarbe("rot");
```

```
        takt = new Taktgeber();
```

```
        takt.setzeID(10);
```

```
        takt.setzeLink( this );
```

```
        takt.endlos();
```

**Erzeugen eines Taktobjekts
Das Taktobjekt erhält die ID 10
Setzen eines Links auf sich selbst**

```
    } // Ende Konstruktor
```

```
    public void bewegen() {
```

```
        lampe.nachRechtsBewegen();
```

```
    } // Ende bewegen
```

```
    public void tuWas(int ID) {
```

```
        bewegen();
```

```
    } // Ende tuWas
```

Die Methode tuWas

```
} // Ende Klasse Bewegung
```

Die Erzeugung des Objekts Takt erfolgt innerhalb der Klasse Bewegung.

Takt soll an das erzeugende Objekt signalisieren, das takt erzeugt.

Daher die Zielangabe this.