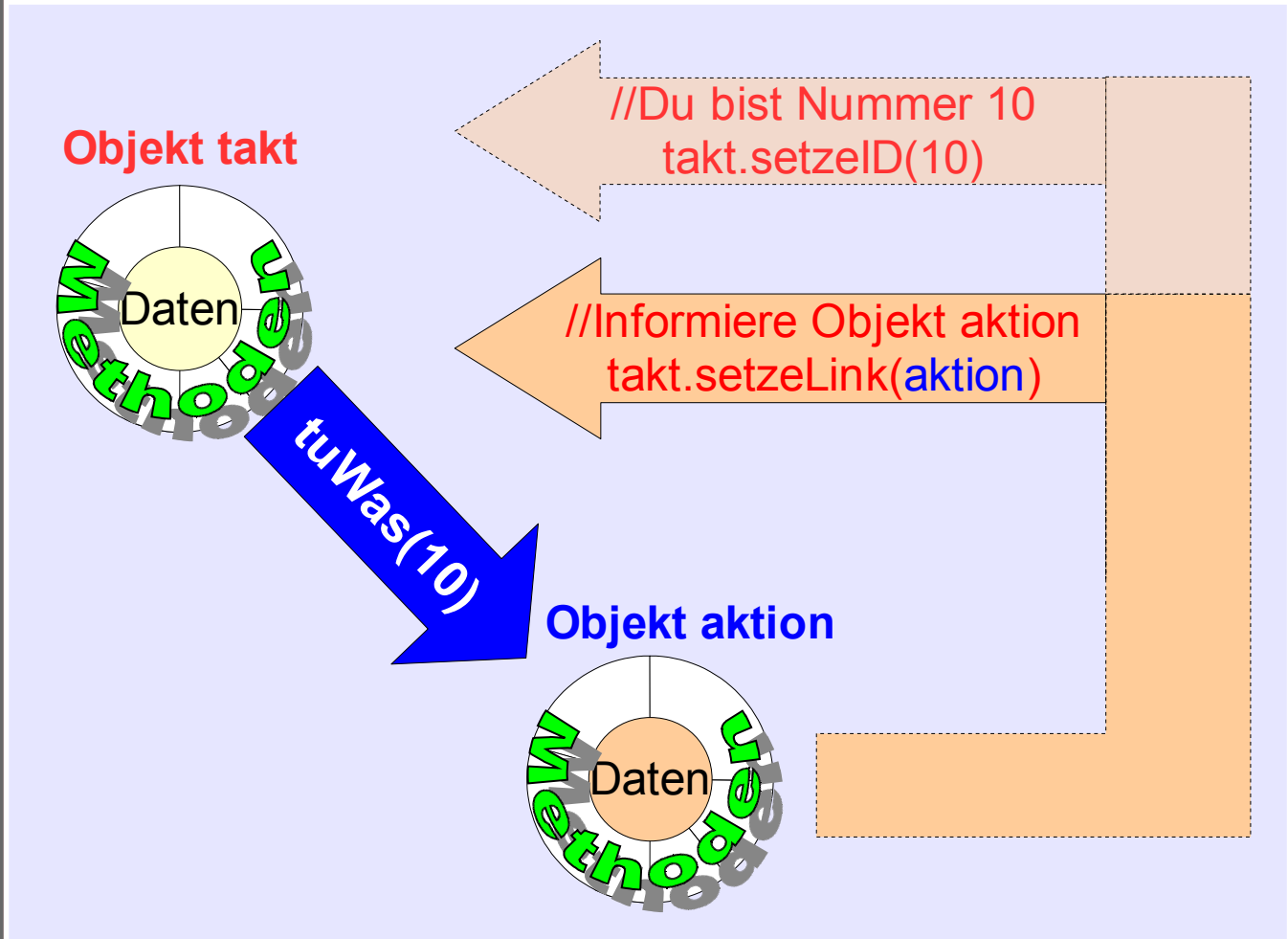


Kommunikation durch Interface oder durch Innere Klassen - eine Gegenüberstellung

Interaktion in Java am Beispiel der Klasse Taktgeber:

Objekte kommunizieren in Java über **Methoden**.

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



[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)

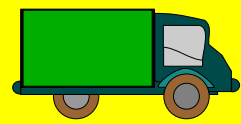
[2] Das Objekt **takt** informiert das Objekt **aktion** durch Aufruf der Methode **tuWas(10)**.

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

Deshalb **implements ITuWAS** im Klassenkopf der dazugehörenden

Klasse **Bewegung**.

(Ist die Initialisierung innerhalb der Klasse des Objekt **aktion**, so wird an **takt** als Adresse **this** übergeben!)



Kommunikation über Interface:

```
/**
```

```
 * 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.setzeLink( this );
```

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

**Erzeugen eines Taktobjekts
Setzen eines Links auf sich selbst**

```
    } // Ende Konstruktor
```

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

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

```
    } // Ende bewegen
```

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

```
        bewegen();
```

```
    } // Ende tuWas
```

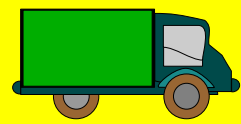
Die Methode tuWas

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

Wissen:

Durch **implements ITuWas** garantiert **Java**, dass die Klasse die Methode mit der Signatur **public void tuWas(int ID)** besitzt.

Das Objekt **takt** kann darauf vertrauen, dass diese Methode existiert.



Kommunikation über innere Klassen:

```
public class Bewegung {
    // Variablen und Objekte
    private Kreis lampe ;
    private Taktgeber takt ;
    /**
     * Konstruktor für Objekte der Klasse Bewegung
     */
    public Bewegung()
    {
        // Objekte initialisieren
        lampe = new Kreis(100, 100,50);
        lampe.setzeFarbe("rot");
        takt = new Taktgeber();
        TuWas link = new ITuWas() {
            public void tuWas(int ID) {
                bewegen(); // Die lokale Methode
            }
        };
        takt.setzeLink(link);
        takt.endlos();
    } // Ende Konstruktor

    public void bewegen() {
        lampe.nachRechtsBewegen();
    } // Ende bewegen
} // Ende Klasse Bewegung
```

Deklaration des Objekts takt

Erzeugen eines Taktobjekts

Die innere Klasse

Die **innere Klasse** vom „Typ“ **ITuWas** hat als einzigen Inhalt die **Methode tuWas()**. Sie ruft zur Bearbeitung der Aktion die Methode **bewegen()** der Klasse auf. Als innere Klasse hat sie Zugriff auf alles innerhalb der Klasse.

Vorteil der inneren Klasse: Bei vielen Komponenten muss nicht über die Abfrage der ID die Komponente identifiziert werden.

Nachteil: Die Konstruktion ist Anfängern schwer vermittelbar.

Eleganter, aber noch unverständlicher für Anfänger, wird es, wenn man die innere Klasse direkt in der Methode erzeuge:

```
setzeLink( new ITuWas() {
    public void tuWas(int ID) {
        bewegen(); // Die lokale Methode
    }
})
```

Viel Vergnügen beim Helfen, wenn eine Klammer fehlt!