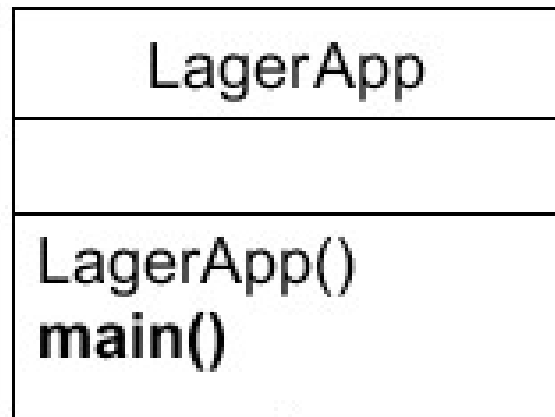
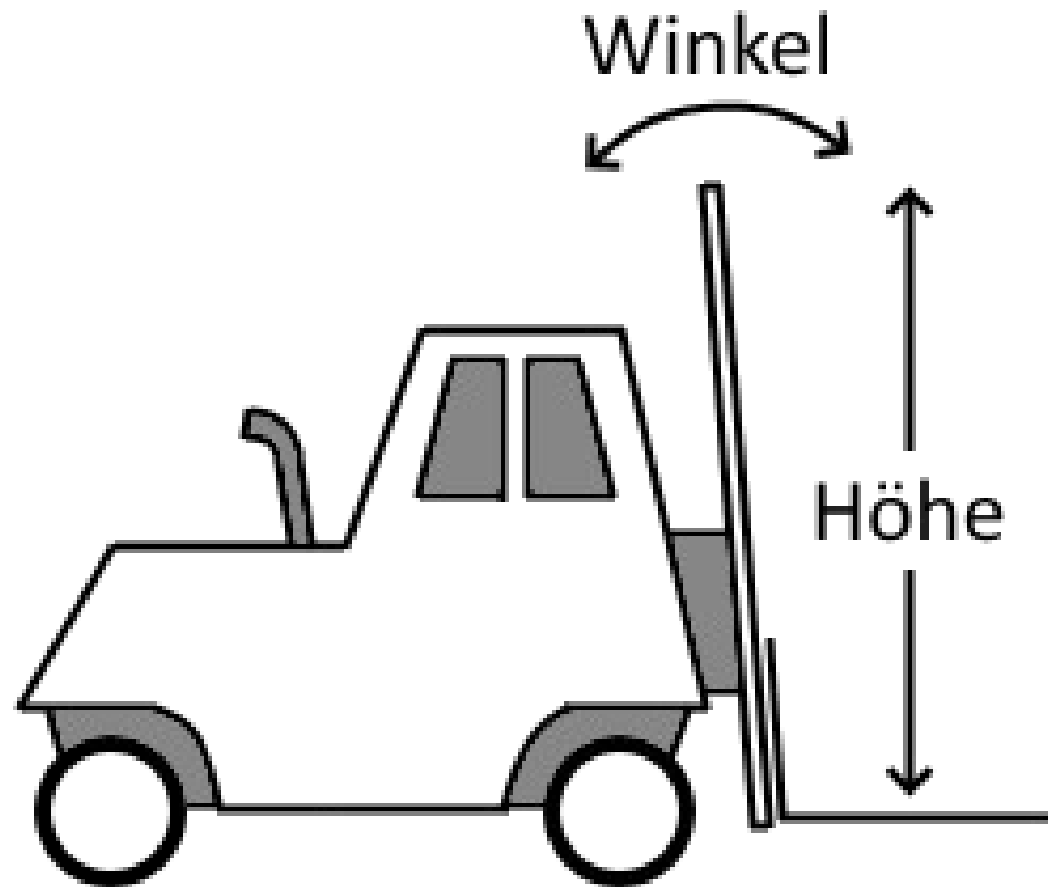
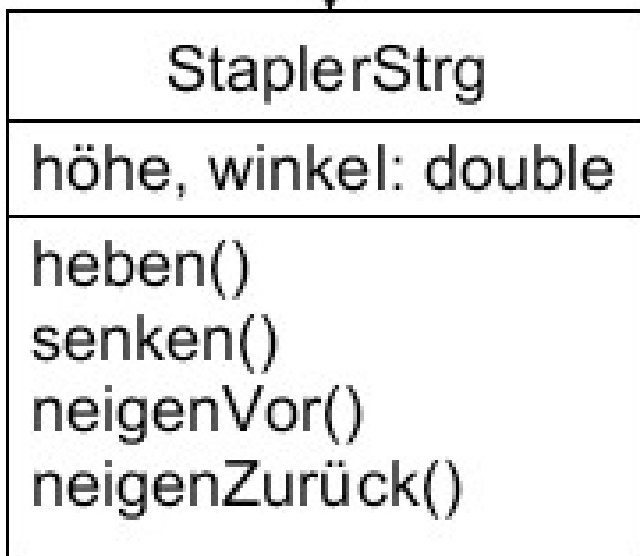


Parameter Einstieg

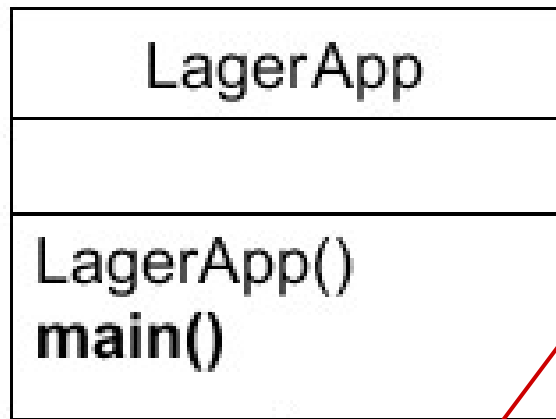
Erinnerung: Staplersteuerung



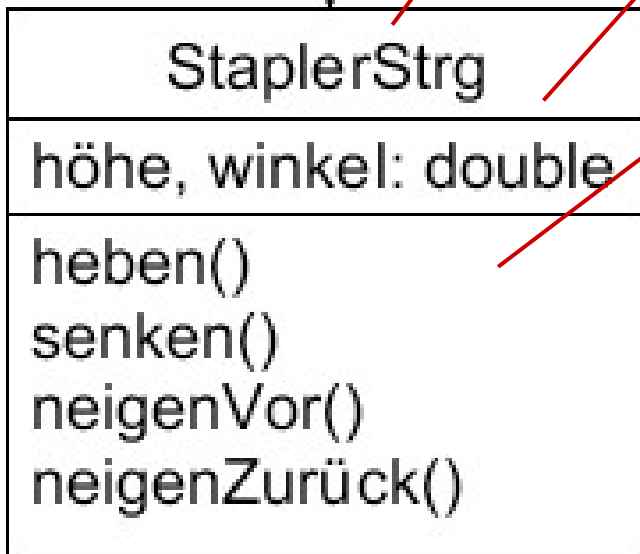
st1, st2



Erinnerung: Implementierung



st1, st2



```
class StaplerStrg  
{
```

```
    double höhe, winkel;
```

```
    ...
```

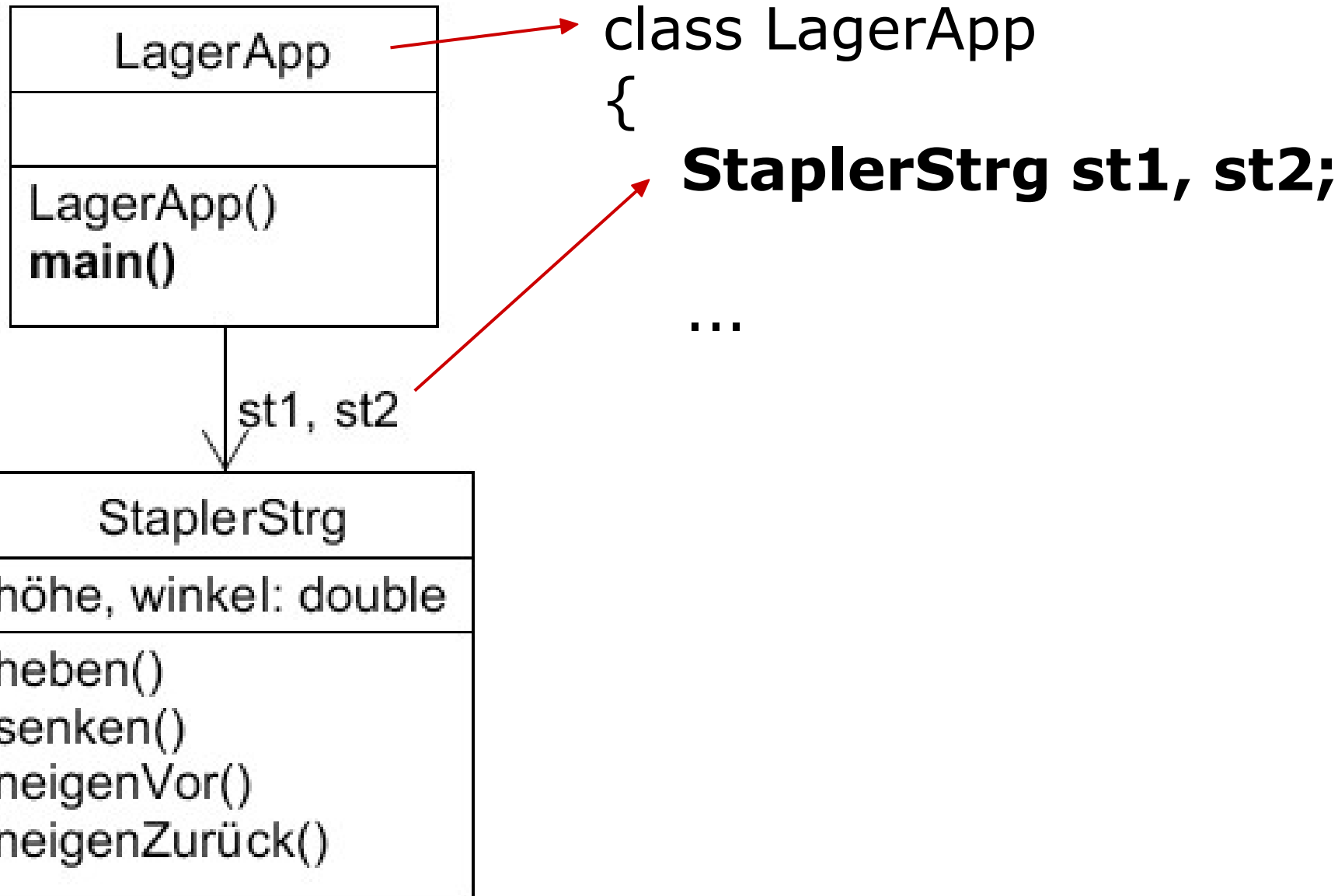
```
    void heben()  
    {
```

```
        höhe = höhe + 10.0;
```

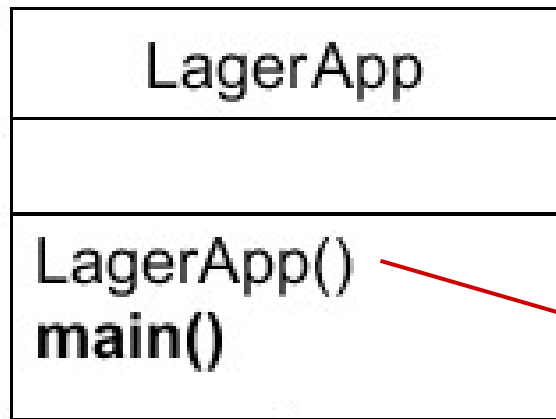
```
    }
```

```
    ...
```

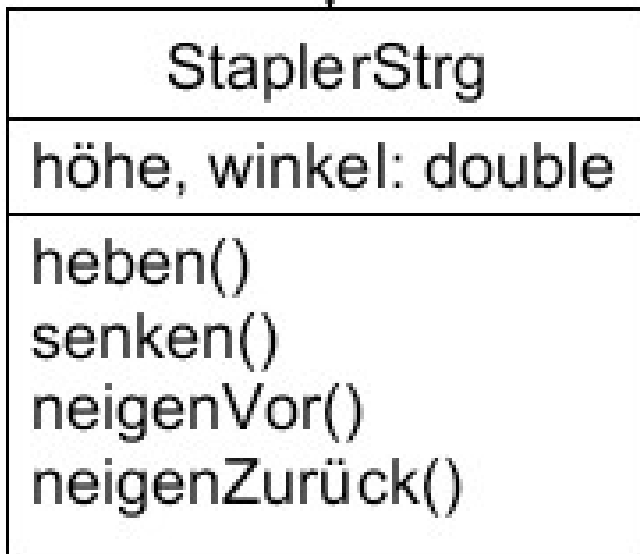
Implementierung der Hauptklasse



Implementierung der Hauptklasse



st1, st2



```
class LagerApp  
{
```

```
    StaplerStrg st1, st2;
```

```
LagerApp()  
{
```

```
    st1 = new StaplerStrg();
```

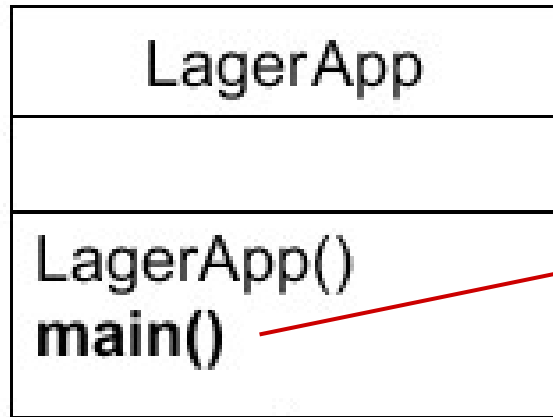
```
    st2 = new StaplerStrg();
```

```
}
```

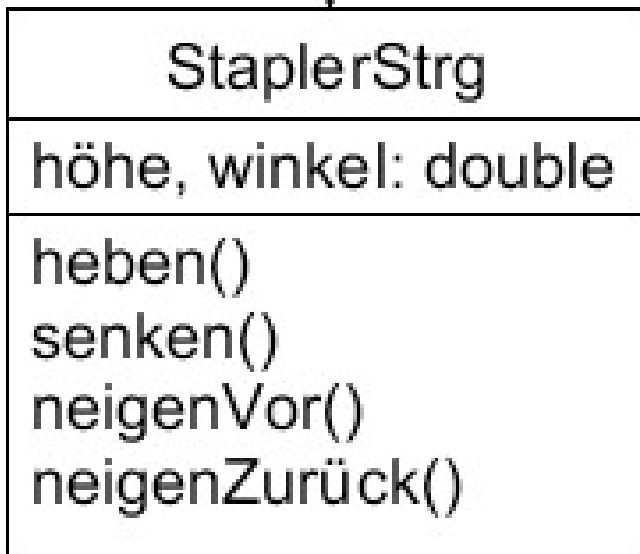
```
...
```

Erinnerung: Implementierung

... (weiter: class Lagerapp) ...



st1, st2



void main()

{

st1.heben();

st1.heben();

st1.heben();

st1.heben();

st1.neigenVor();

st1.neigenVor();

st2.senken();

...

}

Mehrfacher Methodenaufruf ?

„heben“ 4x aufrufen
für 40 cm Bewegung
ist **umständlich**.

```
void main()
{
    st1.heben();
    st1.heben();
    st1.heben();
    st1.heben();
    st1.neigenVor();
    st1.neigenVor();

    st2.senken();
    ...
}
```

Mehrfacher Methodenaufruf ?

Wunsch:
Der Methode heben()
„mitgeben“, **wie weit**
sich die Gabel heben
soll.

```
void main()
{
    st1.heben();
    st1.heben();
    st1.heben();
    st1.heben();
    st1.neigenVor();
    st1.neigenVor();

    st2.senken();
    ...
}
```


Mehrfacher Methodenaufruf ?

Wunsch:

Der Methode heben()
„mitgeben“, **wie weit**
sich die Gabel heben
soll.

```
void main()  
{
```

```
    st1.heben(40);
```

```
    st1.neigenVor();  
    st1.neigenVor();
```

```
    st2.senken();
```

```
    ...
```

```
}
```

Mehrfacher Methodenaufruf ?

(ebenso für die
anderen Methoden)

```
void main()  
{
```

```
    st1.heben(40);
```

```
    st1.neigenVor(10);
```

```
    st2.senken(25);
```

```
    ...
```

```
}
```

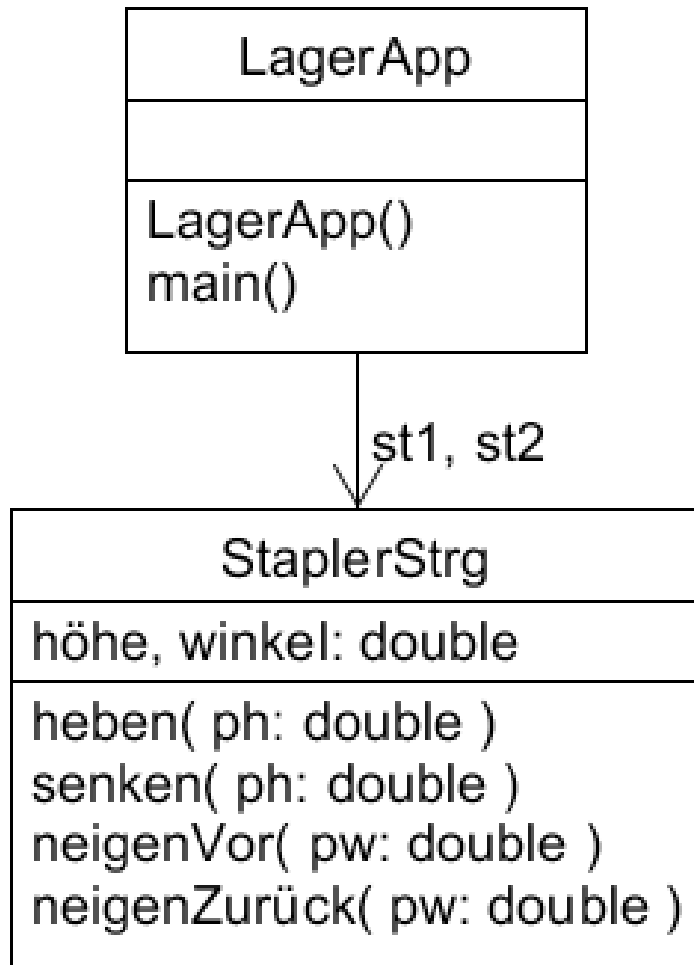
Parameter

Man kann Methoden
Werte „mitgeben“
über **Parameter**:

Beim Aufruf der
Methode schreibt man
einen Wert für den
Parameter in die
Klammern.

```
void main()  
{  
  
    st1.heben(40);  
  
    st1.neigenVor(10);  
  
    st2.senken(25);  
    ...  
}
```

Parameter deklarieren

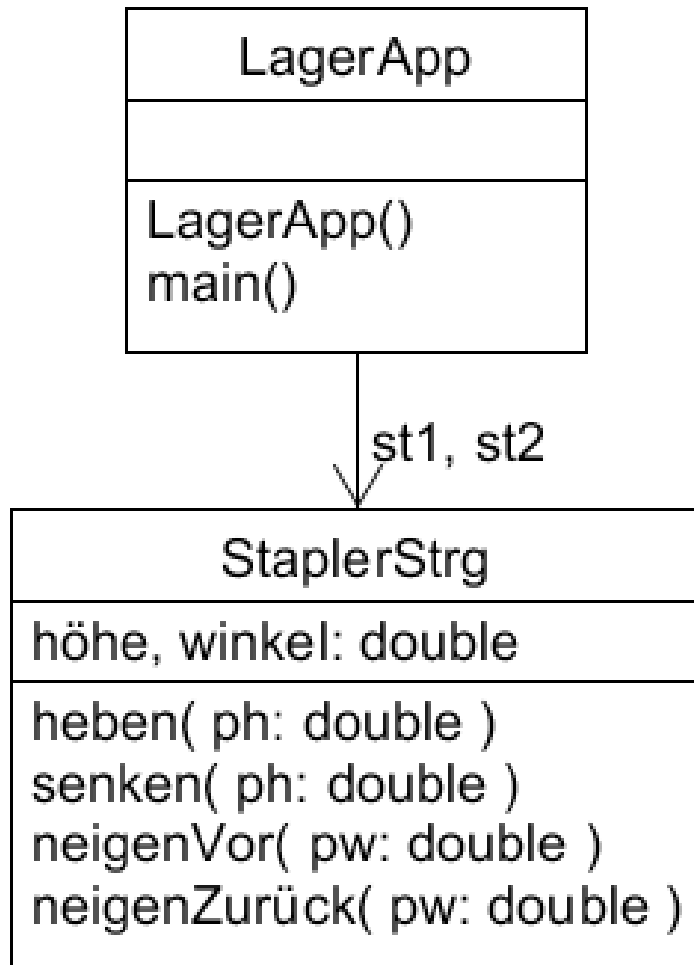


Um Parameter einzusetzen, muss eine Methode einen (oder mehrere) Parameter deklarieren.

Im **Klassendiagramm** schreibt man Parameter in die Klammern der Methoden.

Parameter haben Datentypen (wie Attribute).

Parameter im Quellcode



```
class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }
}
```

Parameter im Quellcode

```
class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }
}
```

```
class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }
}
```

```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }
}

```

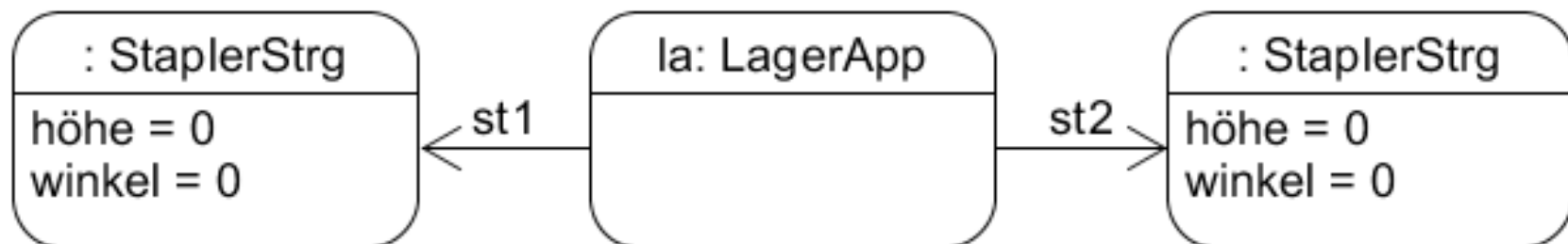
```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }
}

```




```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

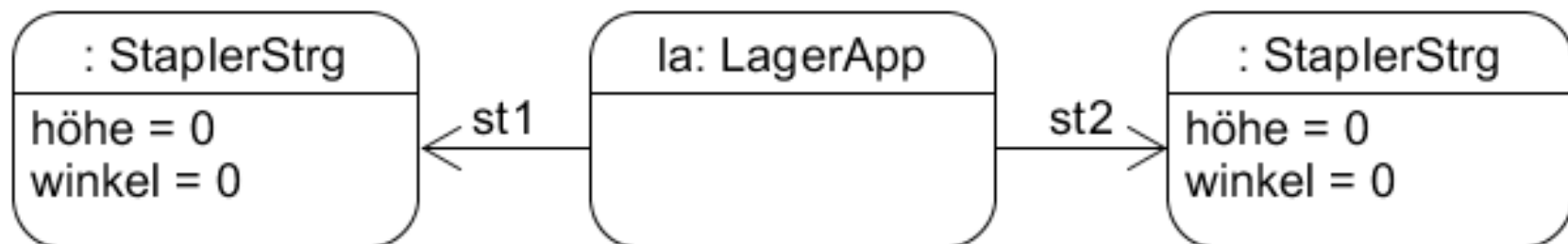
```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        40 ← 0 + 40
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

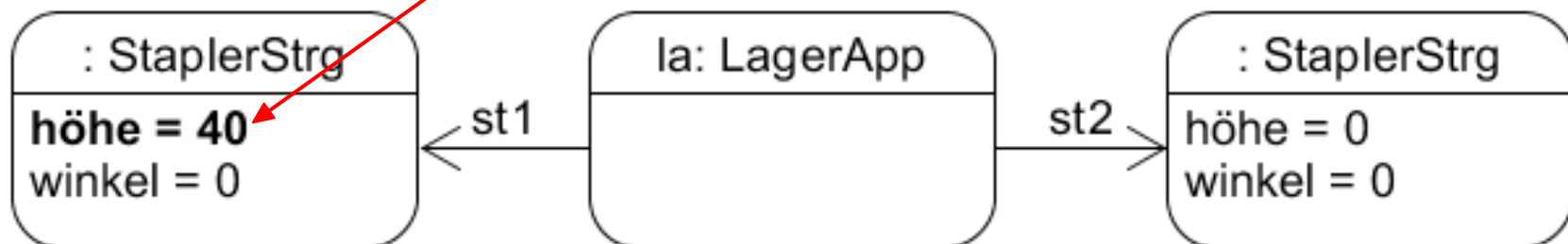
```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        40
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }
}

```

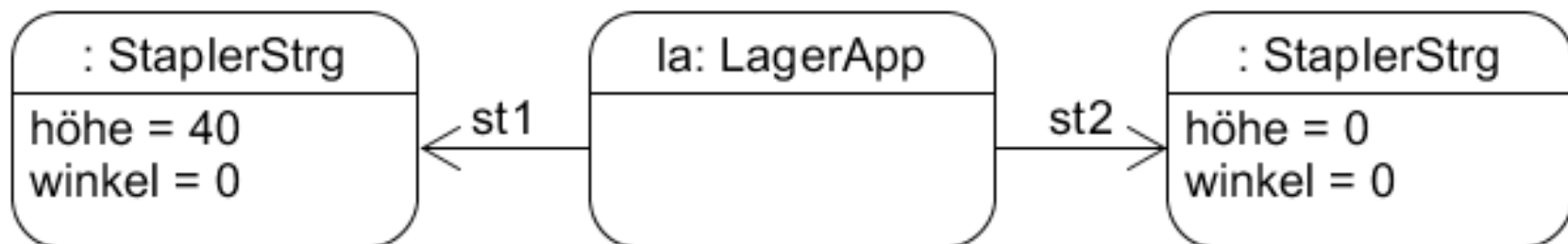
```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }
}

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

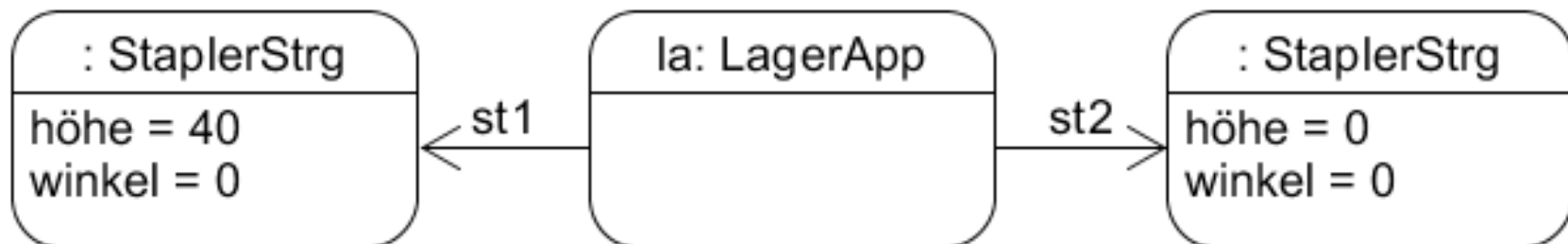
```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        25 ← 0 + 25
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



```

class LagerApp
{
    StaperStrg st1, st2;

    void main()
    {
        st1.heben(40);
        st1.neigenVor(10);

        st2.heben(25);
        ...
    }

```

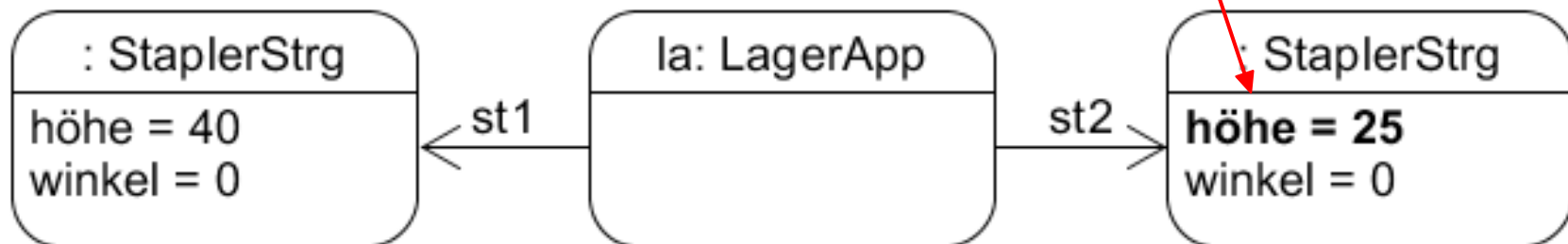
```

class StaplerStrg
{
    double höhe, winkel;

    void heben(double ph)
    {
        25
        höhe = höhe + ph;
    }

    void senken (double ph)
    {
        ...
    }

```



Parameter

- **Deklaration** von Parametern im Methodenkopf:
void <methode> (<datentyp> <param>)

Beispiel: void heben (double ph)

Parameternamen beginnen mit „p“,
um sie von Attributen zu unterscheiden.

Parameter

- Innerhalb einer Methode werden Parameter wie Variablen verwendet. Beispiel:

```
void heben ( double ph )  
{  
    höhe = höhe + ph;  
}
```

Der Parameter existiert nur in dieser Methode !

- Einen **Wert** bekommt ein Parameter erst, wenn die Methode aufgerufen wird. Beispiel:

```
st1.heben ( 40.0 ) ;
```

Autor / Quellen

Autor:

- Christian Pothmann (cpothmann.de)
Freigegeben unter CC BY-NC-SA 4.0, März 2021

