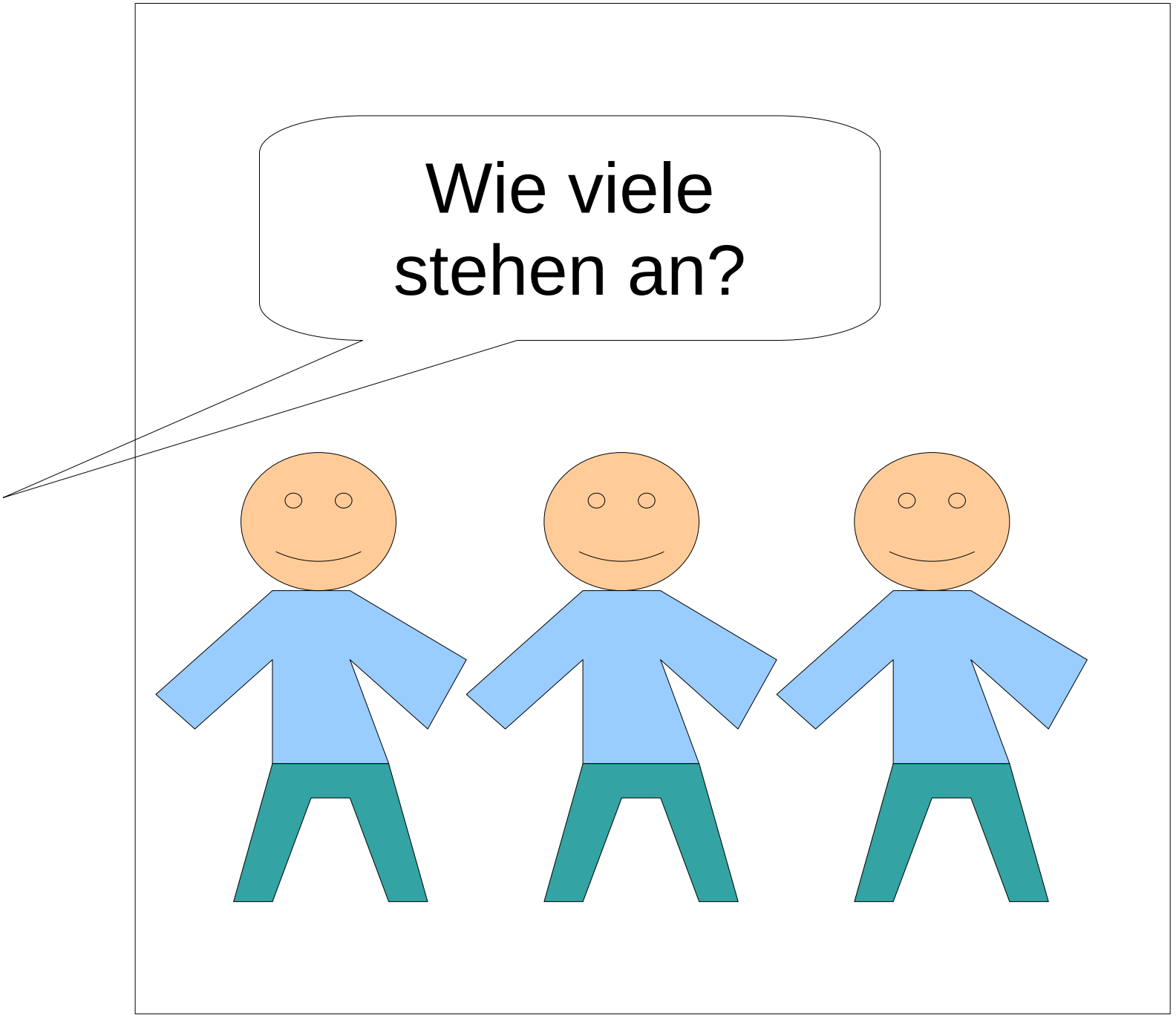
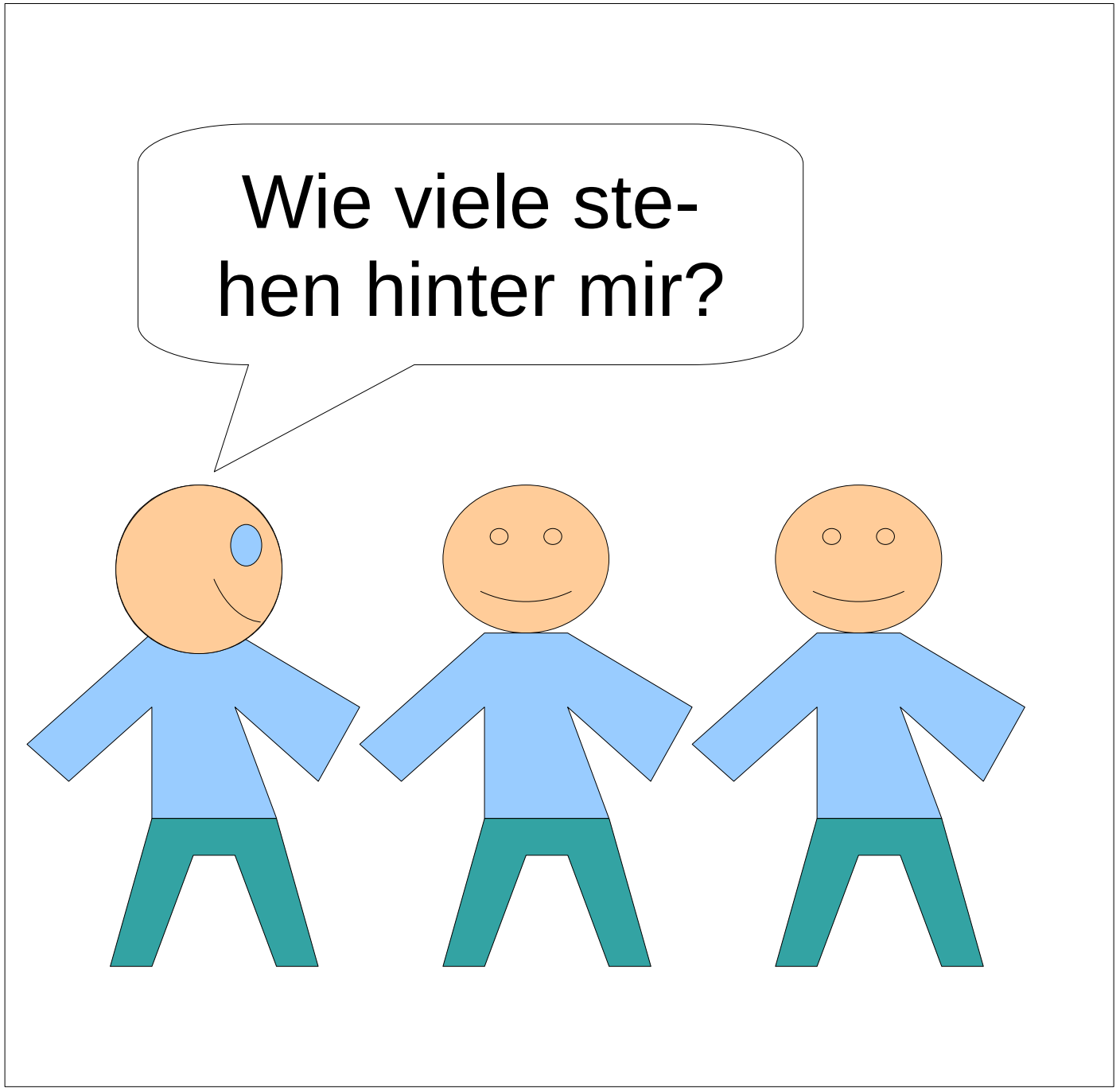
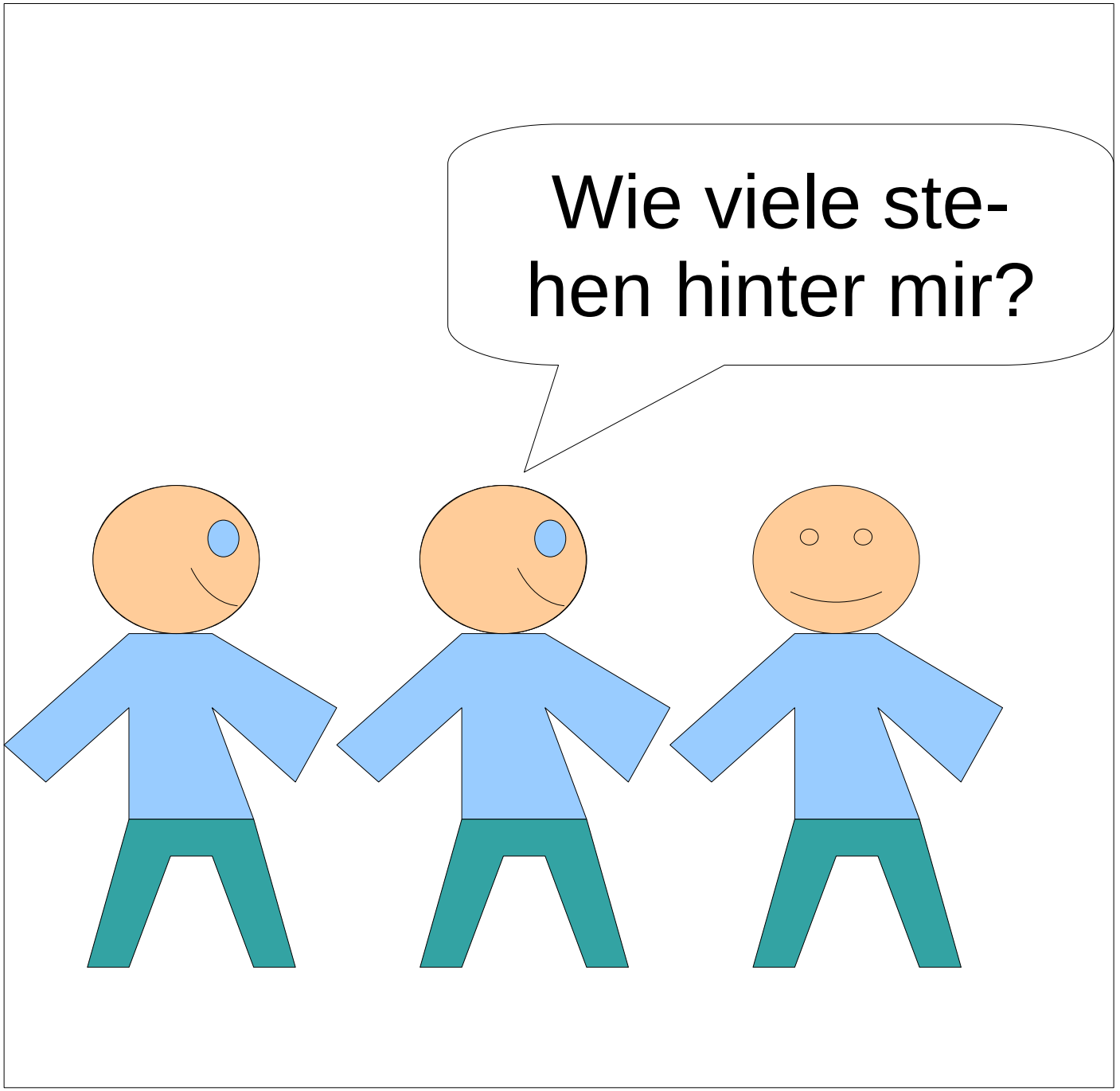
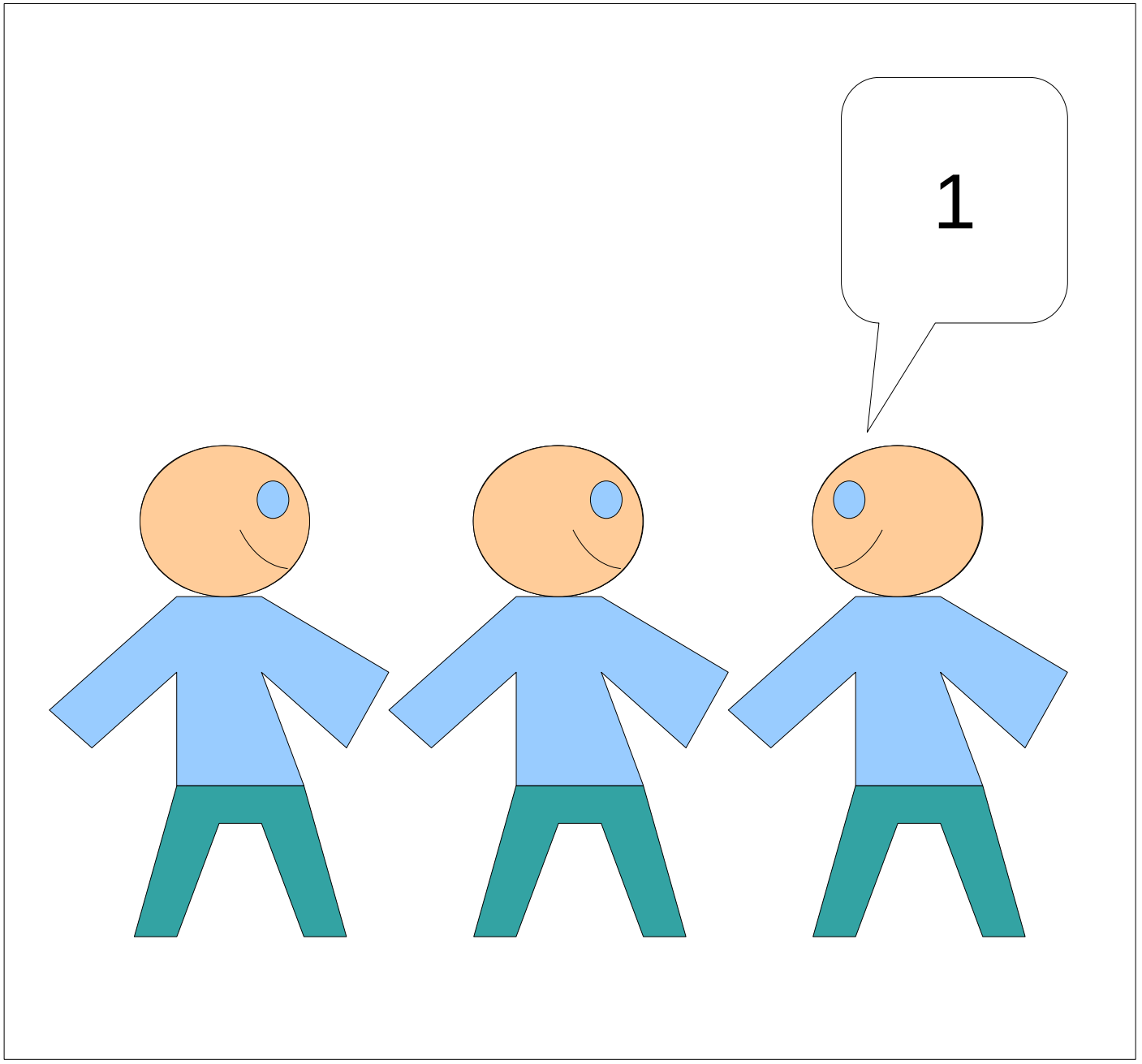
Rekursive Methoden

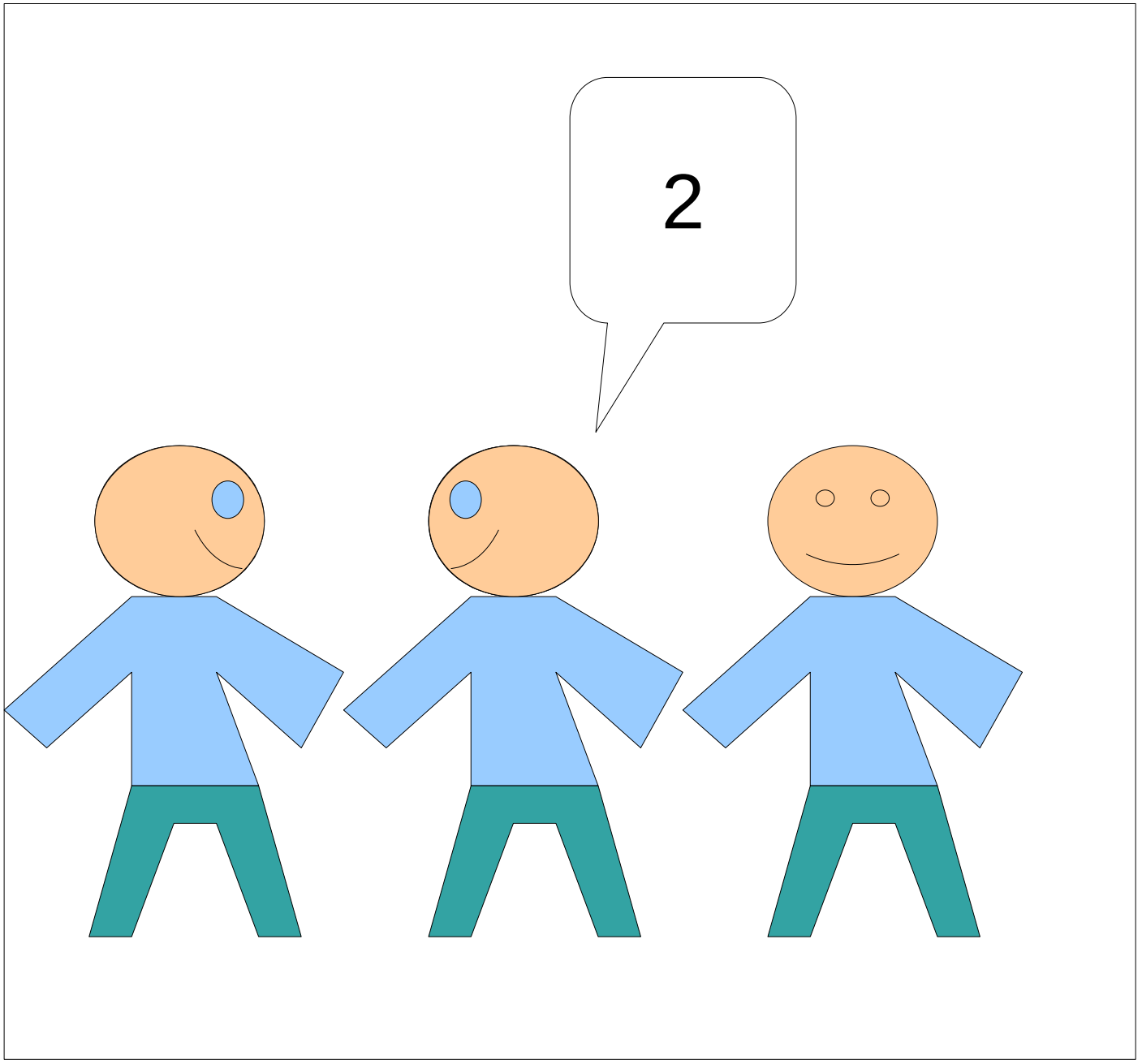
Beispiel: Länge der Liste bestimmen

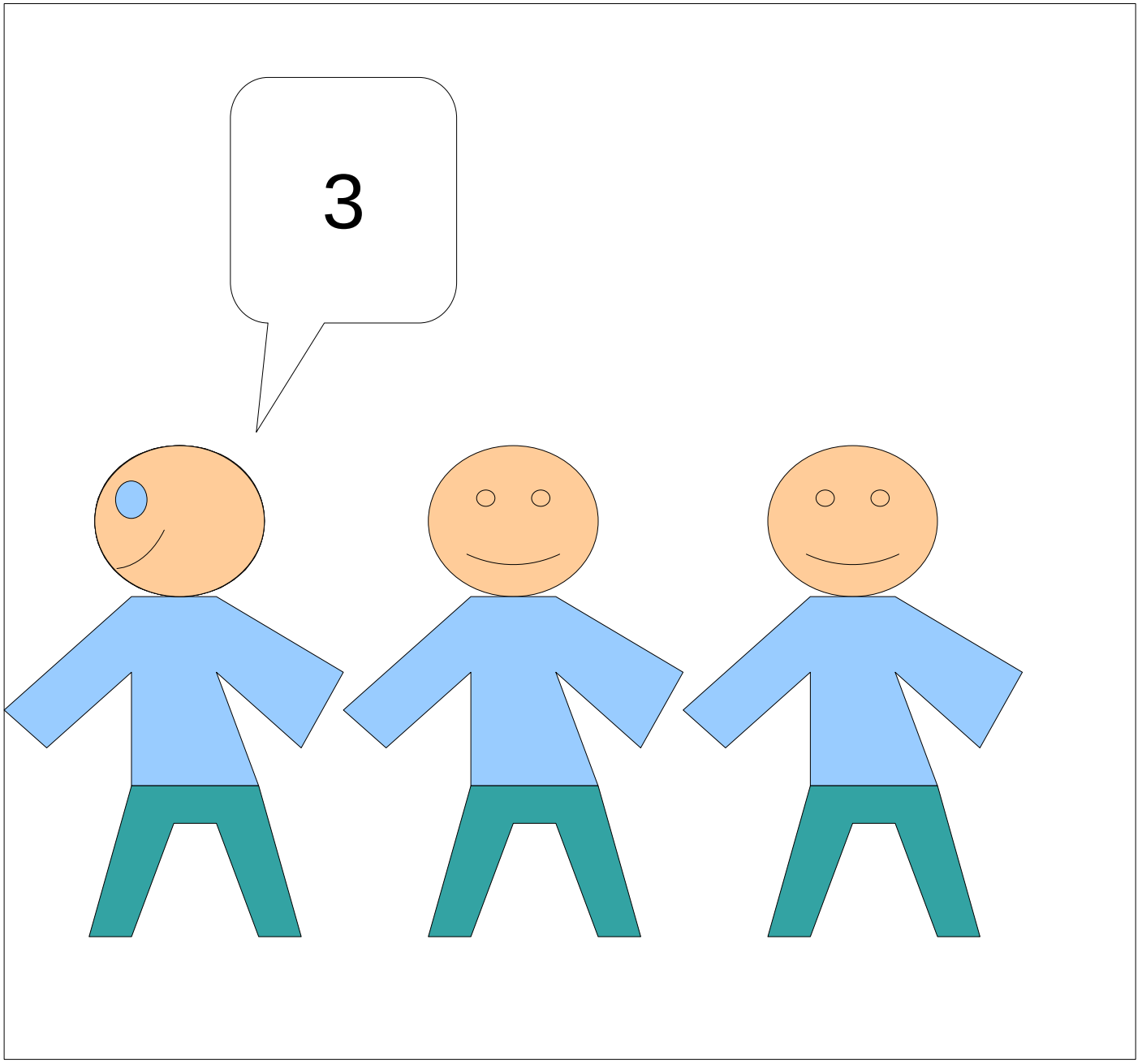

Wie viele
stehen an?


Wie viele ste-
hen hinter mir?


Wie viele ste-
hen hinter mir?


1


2


3

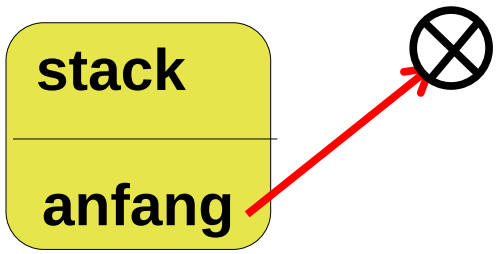
Aufruf der Methode

laenge() des Stapels

1. Fall: Liste leer

Klasse STACK
int laenge()

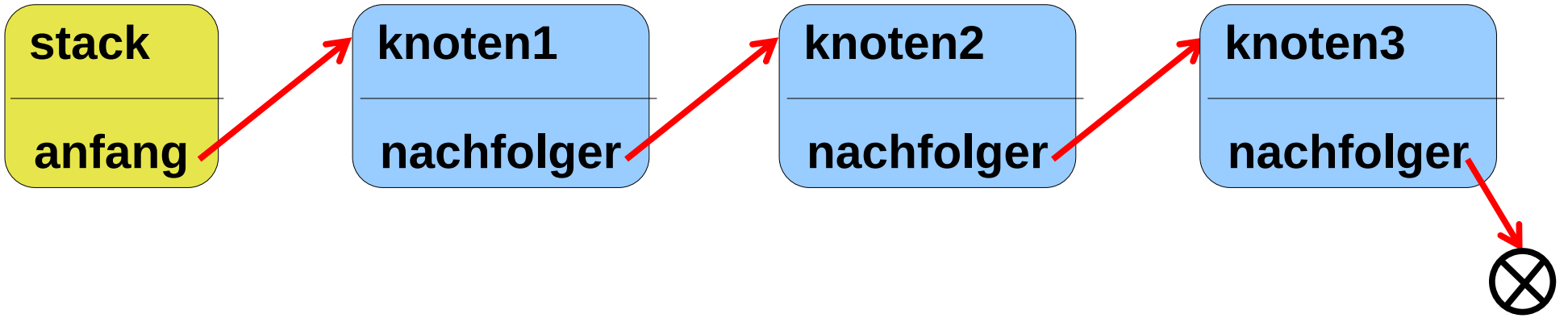
```
wenn(root == null)
dann
    return 0


endewenn
```



2. Fall: Liste nicht leer

Klasse STACK
int laenge()

```
wenn(root == null)
dann
    return 0
sonst
    return root.laenge()
endewenn
```

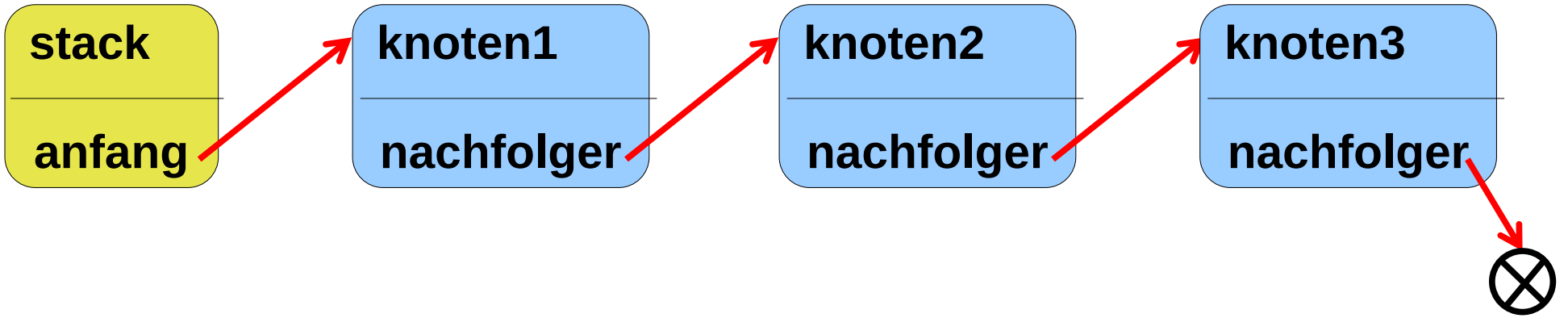


Fall A: Listenende nicht erreicht

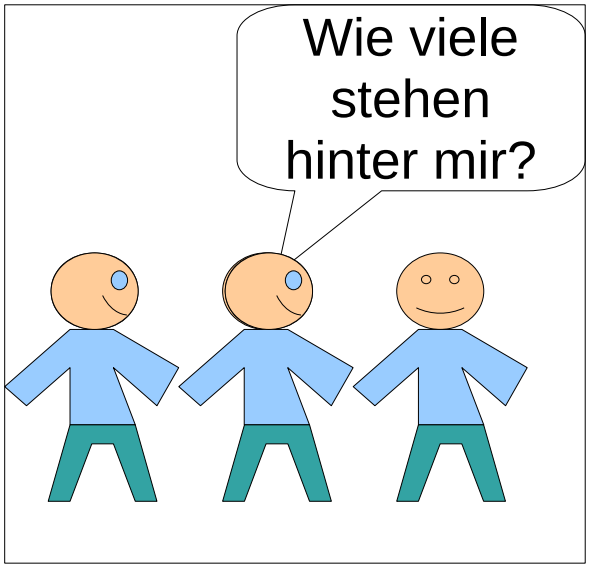
Klasse KNOTEN
int laenge()

```
nachfolger.laenge()
```

Rekursiver Aufruf



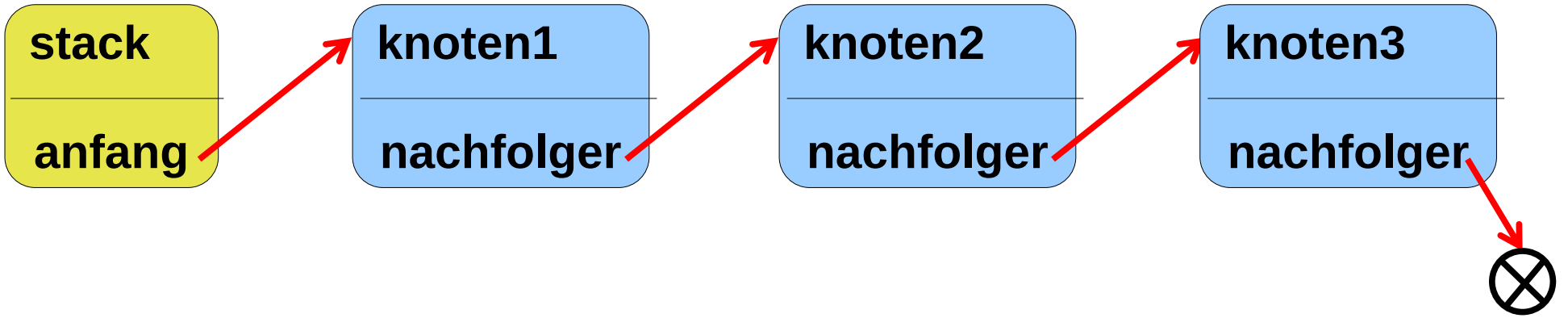
Fall A: Listenende nicht erreicht



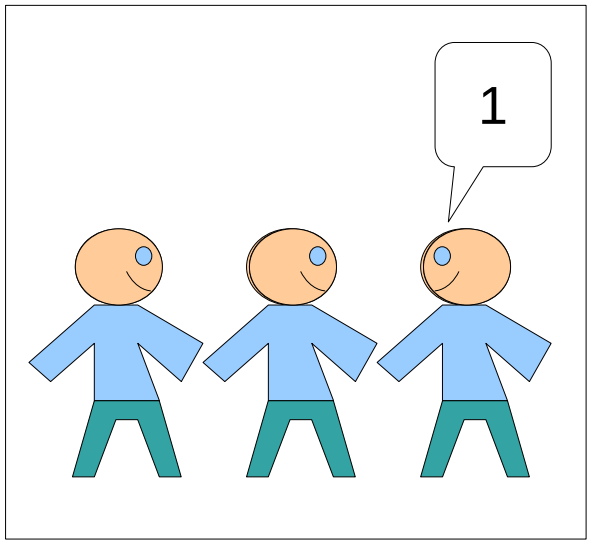
Klasse KNOTEN
int laenge()

```
nachfolger.laenge()
```

Rekursiver Aufruf



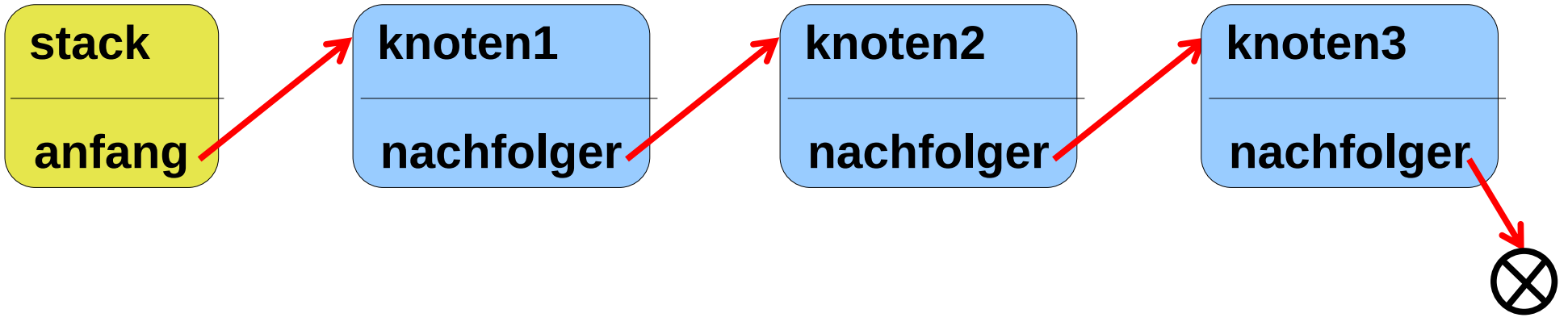
Fall B: Listenende erreicht

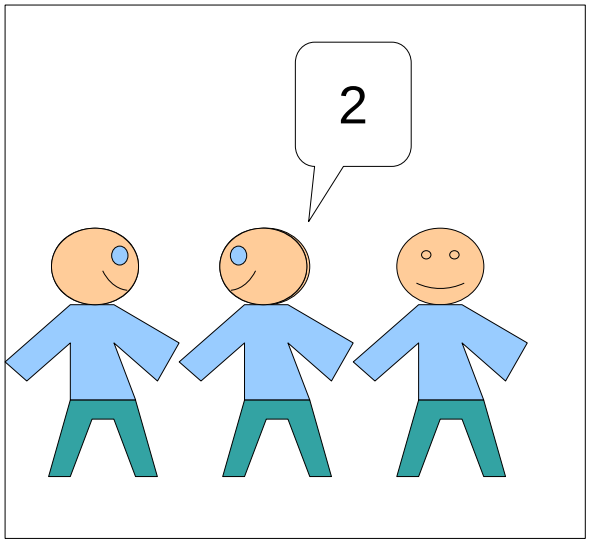


Klasse KNOTEN
int laenge()

```
wenn(nachfolger == null)
dann
    return 1
sonst
        nachfolger.laenge()
endewenn
```

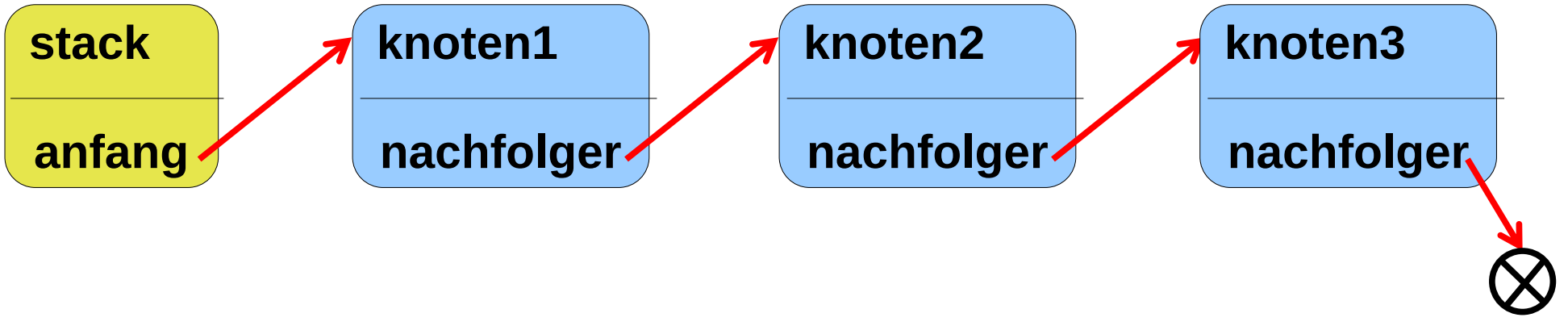
Abbruchbedingung erfüllt

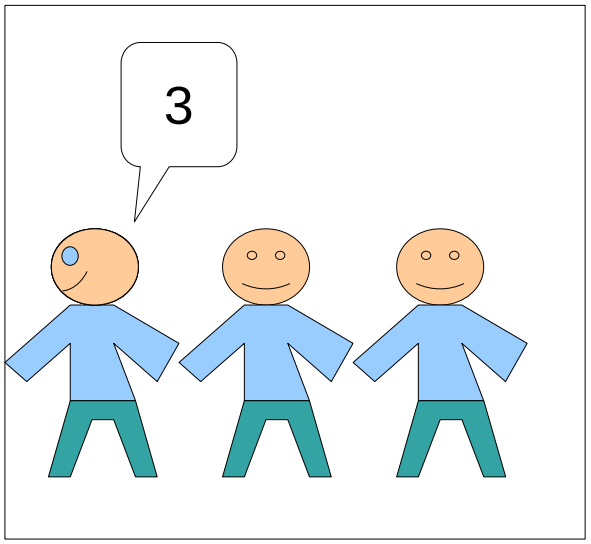




Klasse KNOTEN
int laenge()

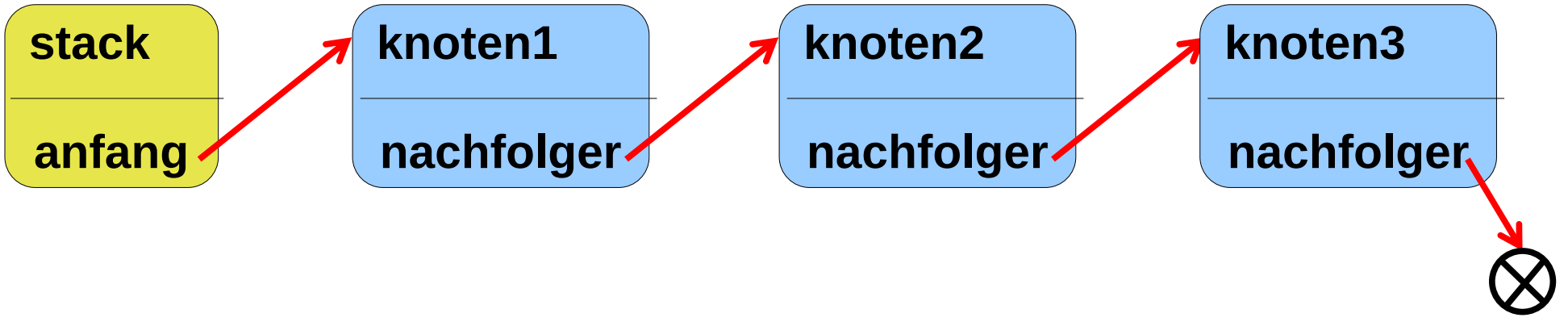
```
wenn(nachfolger == null)
dann
    return 1
sonst
   return nachfolger.laenge()+1
endewenn
```



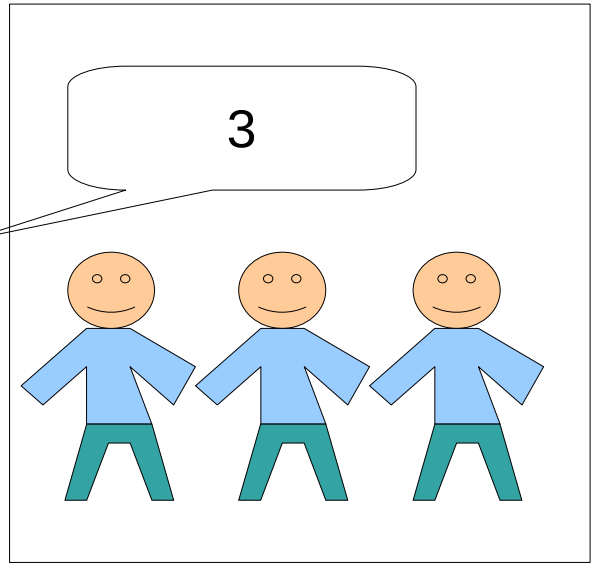


Klasse KNOTEN
int laenge()

```
wenn(nachfolger == null)
dann
    return 1
sonst
   return nachfolger.laenge()+1
endewenn
```



2. Fall: Liste nicht leer



Klasse STACK
int laenge()

```
wenn(anfang == null)
dann
    return 0
sonst
    return anfang.laenge()
endewenn
```