Komposition

Beispiel Gehaltserhöhung… (intelligente Objekte)!

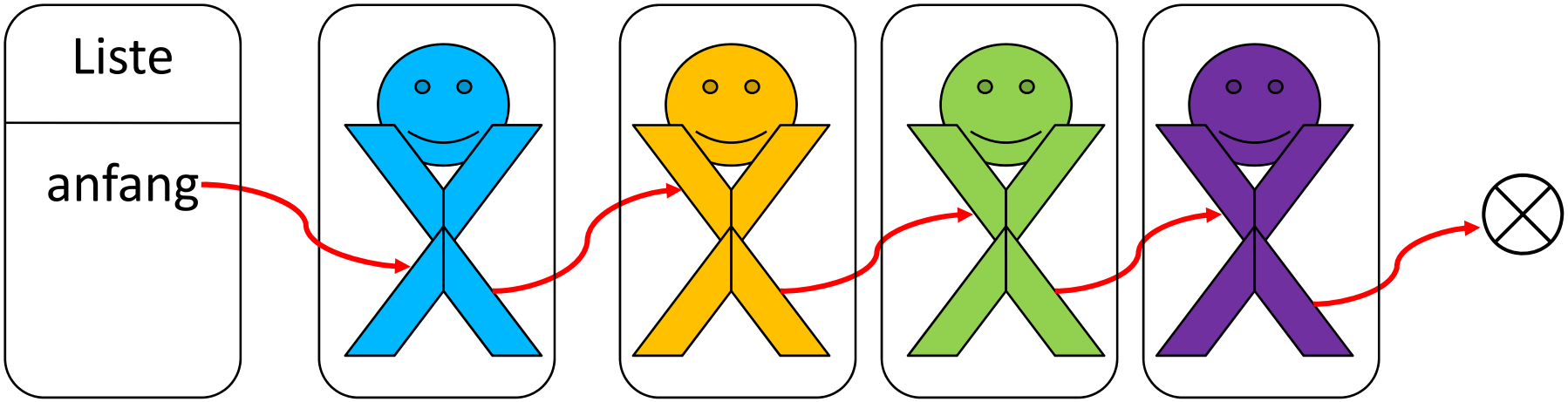
```
public class Mitarbeiter{
  private String name;
  private double gehalt;
…
  public void gehaltserhoehungDurchfuehren(){}
}

public class LeitenderAngestellter extends Mitarbeiter{
  public LeitenderAngestellter(String name, double gehalt){
    super(name, gehalt);
  }
  public void gehaltserhoehungDurchfuehren (){
    grundgehalt=grundgehalt*1.1;
  }
}
public class Sachbearbeiter extends Mitarbeiter{
  …
  public void gehaltserhoehungDurchfuehren (){
    grundgehalt=grundgehalt*1.05;
  }
}
public class Auszubildender extends Mitarbeiter{
  …
  public void gehaltserhoehungDurchfuehren (){
    grundgehalt=grundgehalt+50;
  }
}
```

Listen programmieren als Komposition

Was ist ein Kompositum?

Bisher sah unsere Liste so aus…

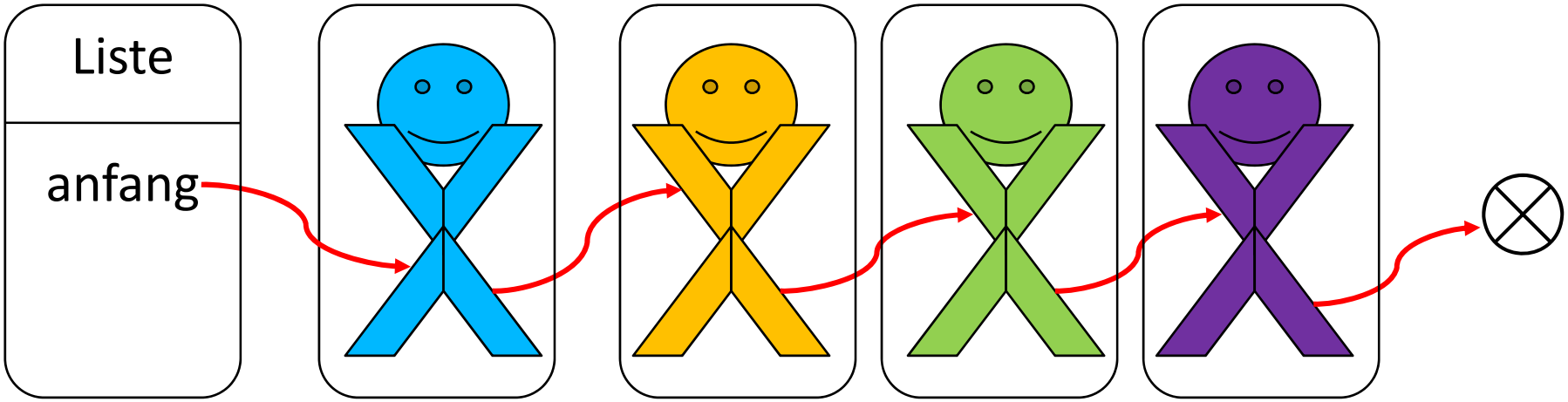


Wie kann man feststellen, ob ein Objekt das letzte Element in der Liste ist?

Listen programmieren als Komposition

Was ist ein Kompositum?

Bisher sah unsere Liste so aus…



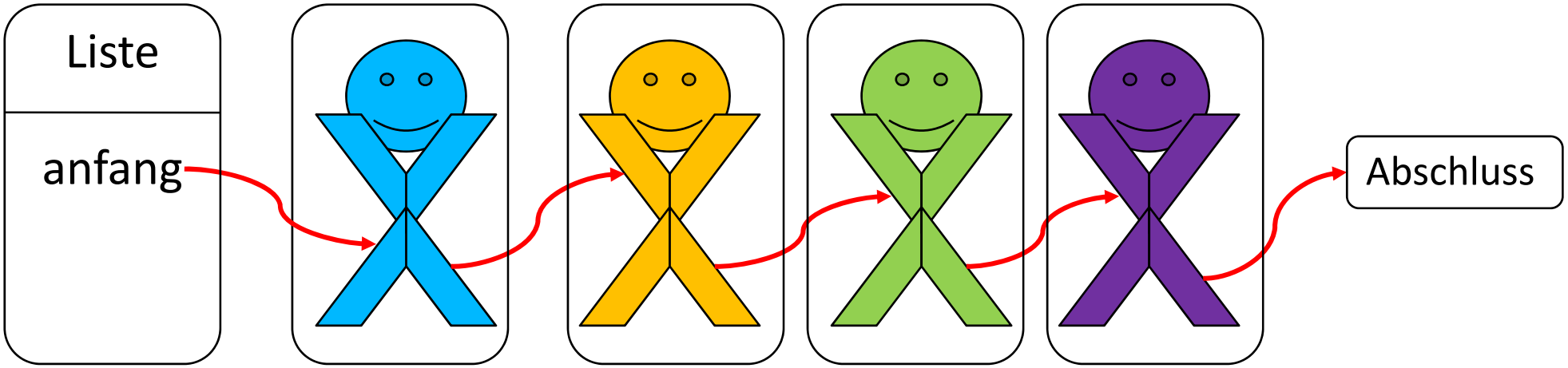
Wie kann man feststellen, ob ein Objekt das letzte Element in der Liste ist?

Wenn es als Nachfolger „null“ hat!

Listen programmieren als Komposition

Was ist ein Kompositum?

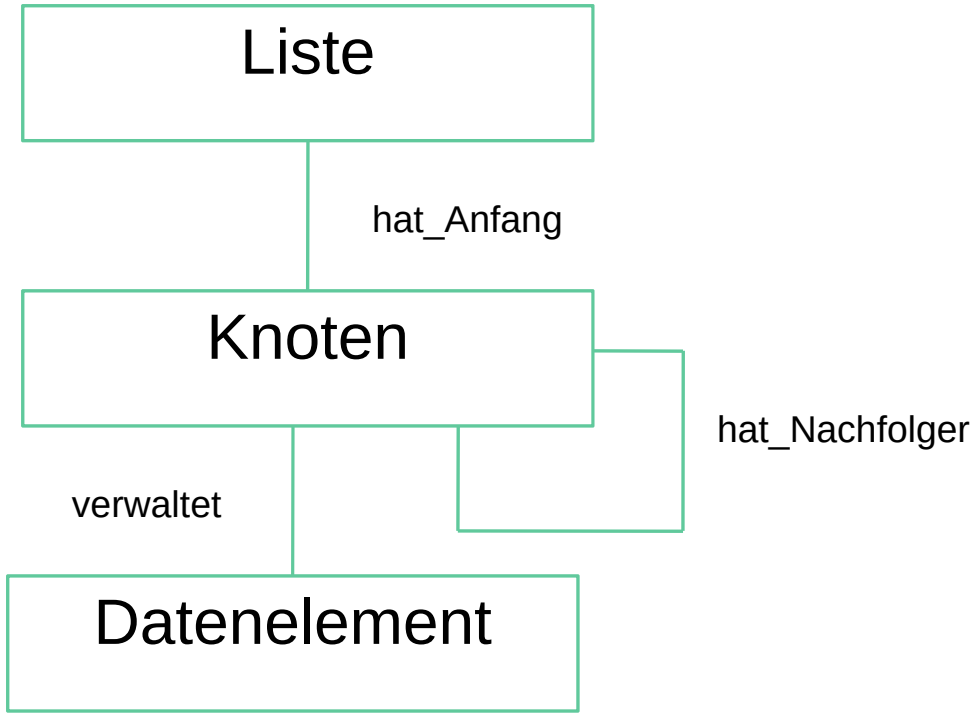
Wir könnten hinten auch ein Objekt als Abschluss nehmen…



Es hat keine Attribute, hilft uns aber, so manchen Code zu vereinfachen!
Schauen wir uns das Klassendiagramm einer Komposition an…

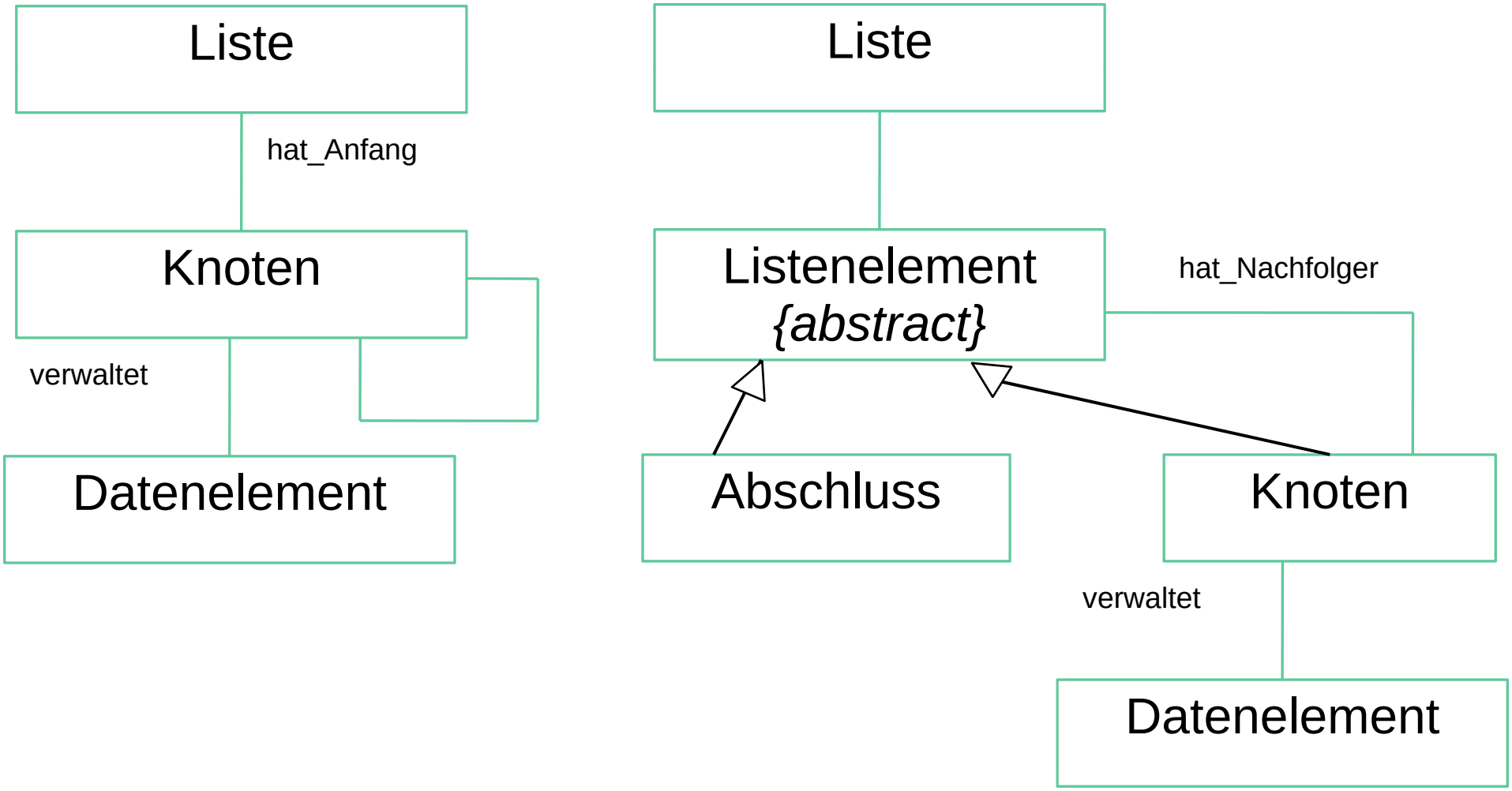
Listen programmieren als Komposition

Was ist ein Kompositum?



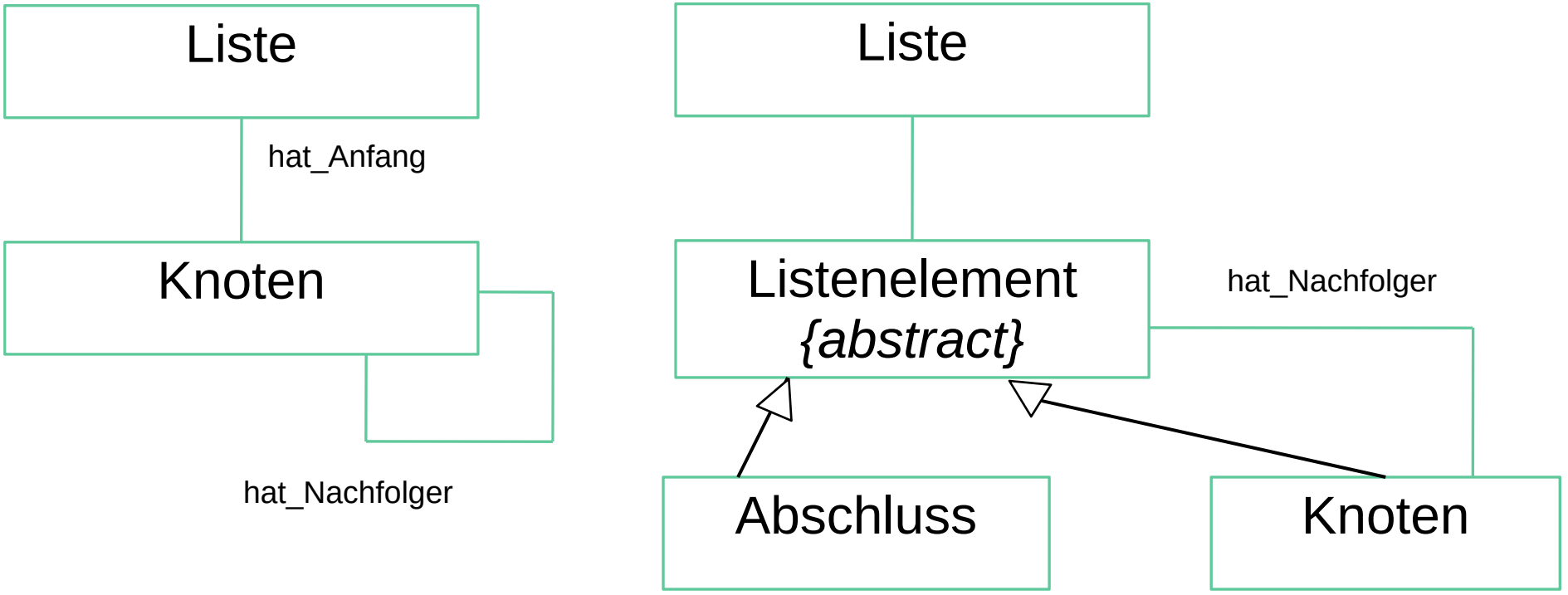
Listen programmieren als Komposition

Was ist ein Kompositum?



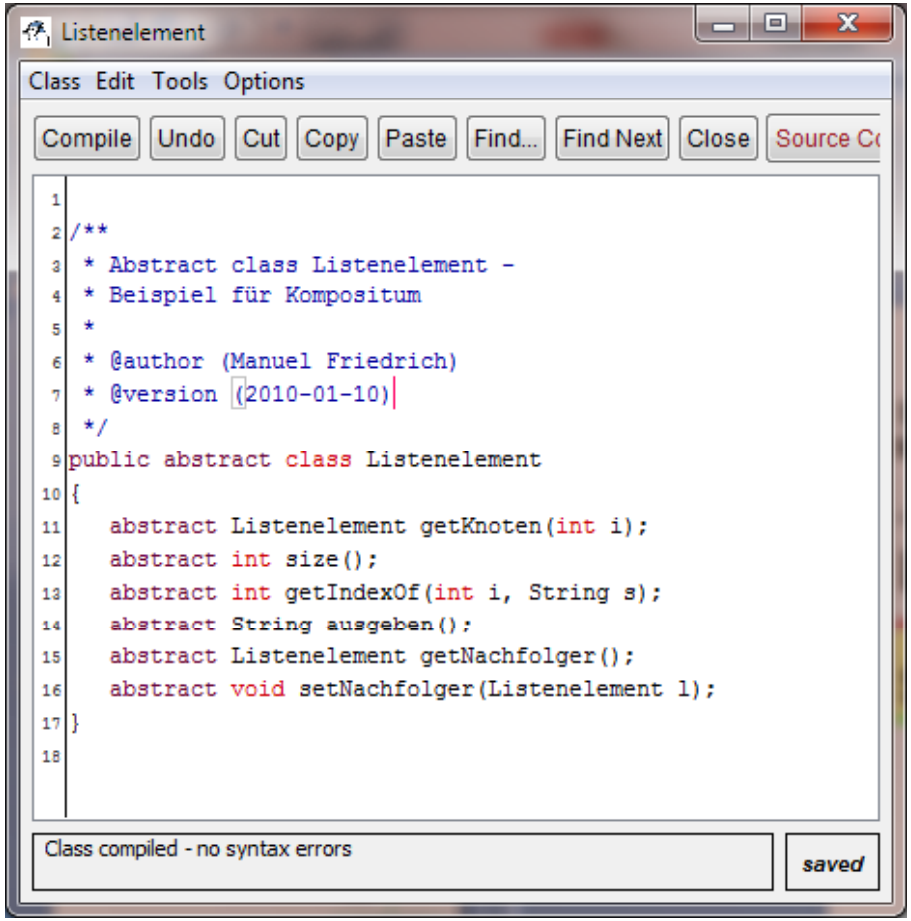
Listen programmieren als Komposition

Was ist ein Kompositum?



Listen programmieren als Komposition

Schauen wir uns die abstrakte Klasse an…



und beschränken uns dann auf die Methode size()

Listen programmieren als Komposition

Vorher…

In der Klasse Liste

```
public int size(){
  if (anfang==null) { return 0;}
else
  { return anfang.size(); }
}
```

In der Klasse Knoten

```
public int size(){
  if (nachfolger==null) { return 1; }
  else
  { return 1+nachfolger.size(); }
}
```

Listen programmieren als Komposition

nachher…

In der Klasse Liste

```
public int size(){
   return anfang.size();
}
```

In der Klasse Knoten

```
public int size(){
      return 1+nachfolger.size();
}
```

In der Klasse Abschluss

```
int size(){return 0;}
```

Listen programmieren als Komposition

Vergleich

In der Klasse Liste

```
public int size(){
   return anfang.size();
}
```

In der Klasse Knoten

```
public int size(){
      return 1+nachfolger.size();
}
```

In der Klasse Abschluss

```
int size(){return 0;}
```

In der Klasse Liste

```
public int size(){
  if (anfang==null) { return
0;}
else
  { return anfang.size(); }
}
```

In der Klasse Knoten

```
public int size(){
  if (nachfolger==null) {
return 1; }
  else
  { return
1+nachfolger.size(); }
}
```

Listen programmieren als Komposition:

Grundwissen:

Wird das Ende einer Liste als Objekt der Klasse Abschluss markiert, gelingt eine besonders einfache Umsetzung der Methoden. Der Aufbau gemäß dem **Entwurfsmuster Kompositum** erhält dabei die folgende Form: