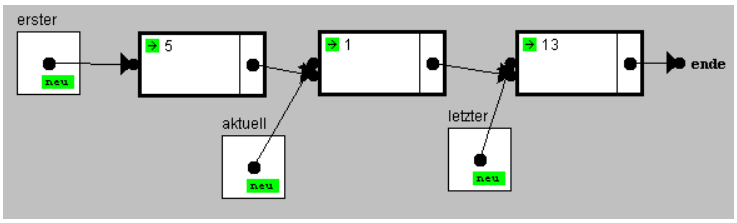
Die Klasse List in Java

Deklaration + Veranschaulichung mit GUI



Die generische Klasse List <ContentType>

```
public class List <ContentType> {
//in protected geändert wegen ListErw (siehe später) - Zugriff erforderlich
   protected class ListNode {
      private ContentType contentObject;
      private ListNode next;

      private ListNode(ContentType pContent) {
         contentObject = pContent;
         next = null;
      }

      public ContentType getContentObject() {
         return contentObject;
      }

      public void setContentObject(ContentType pContent) {
         contentObject = pContent;
      }

      public ListNode getNextNode() {
         return next;
      }

      public void setNextNode(ListNode pNext) {
         next = pNext;
      }
   }
```

Objekte der generischen Klasse **List** verwalten beliebig viele, linear angeordnete Objekte vom Typ **ContentType.** Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden… Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt werden.

*Ein List-Element wird durch die intern deklarierte Klasse **ListNode** festgelegt – es besitzt einen **Inhalts**-Eintrag vom Typ ContentType sowie einen **Verweis** auf das nachfolgende Element (vom Typ ListNode).*

```
   ListNode first;
   ListNode last;
   ListNode current;

   public List() {
      first = null;
      last = null;
      current = null;
   }

   public boolean isEmpty() {
      return first == null;
   }

   public boolean hasAccess() {
      return current != null;
   }
```

```
public void next() { if (hasAccess()) current = current.getNextNode(); }

public void toFirst() { if (!isEmpty()) current = first; }

public void toLast() { if (!isEmpty()) current = last; }

public ContentType getContent() {
   if (hasAccess()) return current.getContentObject(); else return null;
}

public void setContent(ContentType pContent) {
   if (pContent != null && hasAccess()) current.setContentObject(pContent);
}

public void insert(ContentType pContent) {
   if (pContent != null) {
      if (hasAccess()) {
         ListNode newNode = new ListNode(pContent);
         if (current != first) {
            ListNode previous = getPrevious(current);
            newNode.setNextNode(previous.getNextNode());
            previous.setNextNode(newNode);
         } else {
             newNode.setNextNode(first);
             first = newNode;
         }
      } else {
         if (isEmpty()) {
            ListNode newNode = new ListNode(pContent); first = newNode; last = newNode;
         }
      }
   }
}

public void append(ContentType pContent) {
   if (pContent != null) {
      if (isEmpty()) {
         insert(pContent);
      } else {
         ListNode newNode = new ListNode(pContent);
         last.setNextNode(newNode);
         last = newNode;
      }
   }
}
```

```
  public void concat(List<ContentType> pList) {
     if (pList != this && pList != null && !pList.isEmpty()) {
        if (isEmpty()) {
           first = pList.first;
           last = pList.last;
        } else {
           last.setNextNode(pList.first);
           last = pList.last;
        }
        //pList wird gelöscht (nicht in Dokumentation gefordert)
        pList.first = null; pList.last = null; pList.current = null;
     }
  }

  public void remove() {
     if (hasAccess() && !isEmpty()) {
        if (current == first) {
           first = first.getNextNode();
        } else {
          ListNode previous = getPrevious(current);
           if (current == last) last = previous;
           previous.setNextNode(current.getNextNode());
        }
        ListNode temp = current.getNextNode();
        //Element wird gelöscht (nicht gefordert)
        current.setContentObject(null);
        current.setNextNode(null);
        current = temp;
        if (isEmpty()) last = null;
     }
  }

  //Zusatz-Methode - nicht in Dokumentation enthalten,
  //aber sehr hilfreich für andere Methoden!
  //geändert in protected wegen GUI!
  protected ListNode getPrevious(ListNode pNode) {
     if (pNode != null && pNode != first && !isEmpty()) {
        ListNode temp = first;
        while (temp != null && temp.getNextNode() != pNode) temp = temp.getNextNode();
        return temp;
     } else {
        return null;
     }
  }
}
```

Das Projekt “List mit Gui (mit JavaFX)”

```
public class Controller {
   @FXML private TextField tfIsEmpty;  … @FXML private Button btPrevious;

   private List <String> l;

   public void btList_click() { l = new List<String>(); gibAus(); btIsEmpty.setDisable(false); … }
   public void btIsEmpty_click() {
      if (l.isEmpty()) tfIsEmpty.setText("ja - leer!"); else tfIsEmpty.setText("nicht leer!");
   }
   public void btHasAccess_click() {
      if (l.hasAccess()) tfHasAccess.setText("ja - current!"); else tfHasAccess.setText("kein current!");
   }
   public void btNext_click() { l.next(); gibAus(); }
   public void btToFirst_click() { l.toFirst(); gibAus(); }
   public void btToLast_click() { l.toLast(); gibAus(); }
   public void btGetContent_click() { tfGetContent.setText(l.getContent()); }
   public void btSetContent_click() { l.setContent(tfSetContent.getText()); gibAus(); }
   public void btAppend_click() { l.append(tfAppend.getText()); gibAus(); }
   public void btInsert_click() { l.insert(tfInsert.getText()); gibAus(); }
   public void btConcat_click() {
      List <String> l2 = new List <String> (); l2.append("1"); l2.append("2"); l2.append("3");
      l.concat(l2); gibAus();
   }
   public void btRemove_click() { l.remove(); gibAus(); }
   public void btPrevious_click() {
     l.current = l.getPrevious(l.current); gibAus();
   }

  public void gibAus() {
      //Hilfsliste l2 - (nur) für einen Hilfszeiger
      List <String> l2 = new List <String> ();
      l2.current = l.current; //zwischenspeichern
      lv.getItems().clear();
      l.toFirst();
      while (l.hasAccess()) {
         if (l.current == l2.current) {
            lv.getItems().add(l.getContent() + " - current!");
         } else {
            lv.getItems().add(l.getContent());
         }
         l.next();
      }
      l.current = l2.current; //zurücksetzen
   }
}
```