Class List<ContentType>

java.lang.Object
    List<ContentType>

```
public class List<ContentType>
extends java.lang.Object
```

Materialien zu den zentralen NRW-Abiturpruefungen im Fach Informatik ab 2018

Generische Klasse List

Objekt der generischen Klasse List verwalten beliebig viele linear angeordnete Objekte vom Typ ContentType. Auf hoechstens ein Listenobjekt, aktuellesObjekt genannt, kann jeweils zugegriffen werden.
Wenn eine Liste leer ist, vollstaendig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste geloescht wurde, gibt es kein aktuelles Objekt.
Das erste oder das letzte Objekt einer Liste koennen durch einen Auftrag zum aktuellen Objekt gemacht werden.
Ausserdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.
Das aktuelle Objekt kann gelesen, veraendert oder geloescht werden. Ausserdem kann vor dem aktuellen Objekt ein Listenobjekt eingefuegt werden.

Version:

`Generisch_06 2015-10-25`

Author:

`Qualitaets- und UnterstuetzungsAgentur - Landesinstitut fuer Schule`

Field Summary

Fields

Modifier and Type	Field and Description
(package private) List.ListNode	current
(package private) List.ListNode	first
(package private) List.ListNode	last

Constructor Summary

Constructors

Constructor and Description
List() Eine leere Liste wird erzeugt.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	append(ContentType pContent) Falls pContent gleich null ist, geschieht nichts.
void	concat(List<ContentType> pList) Falls es sich bei der Liste und pList um dasselbe Objekt handelt, pList null oder eine leere Liste ist, geschieht nichts.
ContentType	getContent() Falls es ein aktuelles Objekt gibt (hasAccess() == true), wird das aktuelle Objekt zurueckgegeben, andernfalls (hasAccess() == false) gibt die Anfrage den Wert null zurueck.
boolean	hasAccess() Die Anfrage liefert den Wert true, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert false.
void	insert(ContentType pContent) Falls es ein aktuelles Objekt gibt (hasAccess() == true), wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefuegt.
boolean	isEmpty() Die Anfrage liefert den Wert true, wenn die Liste keine Objekte enthaelt, sonst liefert sie den Wert false.
void	next() Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausfuehrung des Auftrags kein aktuelles Objekt, d.h. hasAccess() liefert den Wert false.
void	remove() Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (hasAccess() == false), geschieht nichts.
void	setContent(ContentType pContent) Falls es ein aktuelles Objekt gibt (hasAccess() == true) und pContent ungleich null ist, wird das aktuelle Objekt durch pContent ersetzt.
void	toFirst() Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt.
void	toLast() Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

first

List.ListNode first

last

```
List.ListNode last
```

current

```
List.ListNode current
```

Constructor Detail

List

```
public List()
```

Eine leere Liste wird erzeugt.

Method Detail

isEmpty

```
public boolean isEmpty()
```

Die Anfrage liefert den Wert true, wenn die Liste keine Objekte enthaelt, sonst liefert sie den Wert false.

Returns:

`true, wenn die Liste leer ist, sonst false`

hasAccess

```
public boolean hasAccess()
```

Die Anfrage liefert den Wert true, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert false.

Returns:

`true, falls Zugriff moeglich, sonst false`

next

```
public void next()
```

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausfuehrung des Auftrags kein aktuelles Objekt, d.h. hasAccess() liefert den Wert false.

toFirst

```
public void toFirst()
```

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

toLast

```
public void toLast()
```

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

getContent

```
public ContentType getContent()
```

Falls es ein aktuelles Objekt gibt (hasAccess() == true), wird das aktuelle Objekt zurueckgegeben, andernfalls (hasAccess() == false) gibt die Anfrage den Wert null zurueck.

Returns:

```
das aktuelle Objekt (vom Typ ContentType) oder null, wenn es kein aktuelles Objekt gibt
```

setContent

```
public void setContent(ContentType pContent)
```

Falls es ein aktuelles Objekt gibt (hasAccess() == true) und pContent ungleich null ist, wird das aktuelle Objekt durch pContent ersetzt. Sonst geschieht nichts.

Parameters:

```
pContent - das zu schreibende Objekt vom Typ ContentType
```

insert

```
public void insert(ContentType pContent)
```

Falls es ein aktuelles Objekt gibt (hasAccess() == true), wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefuegt. Das aktuelle Objekt bleibt unveraendert.
Wenn die Liste leer ist, wird pContent in die Liste eingefuegt und es gibt weiterhin kein aktuelles Objekt (hasAccess() == false).
Falls es kein aktuelles Objekt gibt (hasAccess() == false) und die Liste nicht leer ist oder pContent gleich null ist, geschieht nichts.

Parameters:

```
pContent - das einzufuegende Objekt vom Typ ContentType
```

append

```
public void append(ContentType pContent)
```

Falls pContent gleich null ist, geschieht nichts.
Ansonsten wird ein neues Objekt pContent am Ende der Liste eingefuegt. Das aktuelle Objekt bleibt unveraendert.
Wenn die Liste leer ist, wird das Objekt pContent in die Liste eingefuegt und es gibt weiterhin kein aktuelles Objekt (hasAccess() == false).

Parameters:

```
pContent - das anzuhaengende Objekt vom Typ ContentType
```

concat

```
public void concat(List<ContentType> pList)
```

Falls es sich bei der Liste und pList um dasselbe Objekt handelt, pList null oder eine leere Liste ist, geschieht nichts.
Ansonsten wird die Liste pList an die aktuelle Liste angehaengt. Anschliessend wird pList eine leere Liste. Das aktuelle Objekt bleibt unveraendert. Insbesondere bleibt hasAccess identisch.

Parameters:

`pList - die am Ende anzuhaengende Liste vom Typ List`

remove

```
public void remove()
```

Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (hasAccess() == false), geschieht nichts.
Falls es ein aktuelles Objekt gibt (hasAccess() == true), wird das aktuelle Objekt geloescht und das Objekt hinter dem geloeschten Objekt wird zum aktuellen Objekt.
Wird das Objekt, das am Ende der Liste steht, geloescht, gibt es kein aktuelles Objekt mehr.