Table Of Content

Class Elem

```
java.lang.Object
    |
    +--Uebungsaufgabe_1_2.Elem
```

< Constructors > < Methods >

public class **Elem**
extends java.lang.Object

Author:

abk640

Constructors

Elem

```
public  Elem(java.lang.Object obj)
```

Konstruktor zur Erzeugung eines Elements.

Parameters:

obj - = Wert des neuen Elements, kann ein beliebiges Objekt sein.

Methods

equals

```
public boolean equals(java.lang.Object obj)
```

StandardmaeÃŸige equals Methode zur Ueberpruefung auf Wertgleichheit eines Elements und eines beliebigen Objekts.

Parameters:

obj - = Zu ueberpruefendes Obejtk (Any)

Returns:

= Gibt TRUE zurueck, sobald der Inhalt beider Operanden gleich ist, ansonsten FALSE.

Overrides:

equals in class java.lang.Object

next

`public Elem next()`

Gibt das Eelement zurueck, auf das gezeigt wird.

Returns:

= Gibt ein Element zurueck, auf das als naechstes gezeigt wird.

---

obj

`public java.lang.Object obj()`

Getter- Methode zur Rueckgabe des tatsaechlichen Wertes eines Elements.

Returns:

= Gibt den Wert, des aktuelle Elements zurueck.

---

setNext

`public void setNext(Elem e)`

Setz den Zeiger auf das naechste Element auf das uebergebene Element.

Parameters:

e - = Das neue Element, auf das gezeigt werden soll.

---

toString

`public java.lang.String toString()`

StandardmaeÃŸige toString Methode, um ein Element in einen String zu konvertieren.

Returns:

= Gibt das Element in Form eines String zurueck.

Overrides:

toString in class java.lang.Object

Interface IListel

< Methods >

---

public interface **IListel**

Author:

abk640

Methods

cons

```
public void cons(java.lang.Object x)
```

Haengt ein Element x an die vorderste Stelle der Liste und markiert dieses als neues Kopfelement.

Parameters:

x - = Anzuhaengender Wert

---

head

```
public void head()
```

Entfernt das vorderste Element einer Liste.

---

insert

```
public boolean insert(java.lang.Object x,
                      int n)
```

Fuegt an eine spezifische Stelle in der Liste ein Element an. Dazu iteriert es bis an die Stelle, wo er es einsetzen mÃ¶chte und versetze den Fokus, des Pointer, des vorherigen Elements auf das neue Element. Der Fokus des Pointers, des neuen Elements wird auf das nachfolgende Element gesetzt.

Parameters:

x - = Einzufuegendes Element
n - = Stelle, an der das neue Element eingefuegt werden soll

Returns:

= Gibt TRUE zuruekc, sobald das Einfuegen erfolgreich war

---

isEmpty

```
public boolean isEmpty()
```

Prueft, ob die Liste leer ist.

Returns:

= Gibt TRUE zurueck, wenn sie leer, ansonsten FALSE

---

length

`public int` **`length`**`()`

Gibt die Laenge einer Liste zur. Laenge der Liste wird bestimmt durch die, in der Liste gespeicherten Elemente.

Returns:

= Gibt eine ganzzahlige Laenge einer Liste zurueck

Interface Liste

< Methods >

---

public interface **Liste**

Praktikumsgruppe_2 Aufgabe_1

Author:

Fabian Sawatzki, Daniel Glake

Methods

cons

`public void` **`cons`**`(java.lang.Object x)`

Konkateniert ein neues Element an die vorderste Stelle der Liste. Dazu wird ein neues Element erzeugt, in der der Wert abgespeichert wird. Die Referenz vom vorherigen ersten Element wird dann auf das neue Element gesetzt, wÃ¤hrend das neue Element auf das alte zeigt.

Parameters:

x - = Wert des anzuhÃ¤ngenen Elementes

---

equals

```
public boolean equals(java.lang.Object obj)
```

StandardmÃ¤ÃŸige equals Methode, die eine Liste und ein Any Objekt verlgleicht, ob sie vom Inhalt her gleich sind.

Parameters:

obj - = Zu verlgeichendes Objekt.

Returns:

= Gibt TRUE zurÃ¼ck, sobald beide Objekte inhaltlich gleich sind, ansonsten FALSE

Overrides:

equals in class java.lang.Object

---

getKopf

```
public Elem getKopf()
```

Getter- Methode um das vorderste Element zurÃ¼ck zu geben.

Returns:

= Gibt das vorderste Element zurÃ¼ck

---

head

```
public void head()
```

Entfernt das vorderste Element an der List. Dazu entfernt es einfach die Referenz auf das vorderste Element und setzt diese dann auf das nÃ¤chste.

---

insert

```
public boolean insert(java.lang.Object x,
                      int n)
```

FÃ¼gt an einer bestimmten Stelle, der List ein neues Element ein. Dazu iteriert es einmal durch die List und inkrementiert einene Zaehler bis der gewÃ¼nschte Ort gefunden ist. AnschlieÃŸen wird der Zeiger des vorderen Elements auf das neue Element gelenkt, wÃ¤hrend das neue Element auf das hinter zeigt. Ist der gewÃ¼nschte Platz Nr. 0 wird jedoch einfach cons() aufgerufen.

Parameters:

x - = Der Wert, des neuen Elements
n - = Die Stelle, an der das neue Element eingefÃ¼gt werden soll.

Returns:

= Gibt ein TRUE zurÃ¼ck, sobald alles erfolgreich abgeschlossen ist, ansonsten FALSE.

isEmpty

`public boolean isEmpty()`

ÃœberprÃ¼ft, ob an der Kopf Zeiger auf irgendein Element zeigt, dass nicht null ist.

Returns:

= Wenn auf kein Element gezeigt wird dann TRUE, ansonsten FALSE.

---

length

`public int length()`

Iteriert einmal Ã¼ber die gesamte Liste, um die Anzahl von Elementen zu zÃ¤hlen. Hat somit eine Laufzeit von O(n).

Returns:

= Gibt die LÃ¤nge der Liste als Integer zurÃ¼ck.

Class ListeImpl

```
java.lang.Object
    |
    +--Uebungsaufgabe_1_2.ListeImpl
```

All Implemented Interfaces:

Liste

< Constructors > < Methods >

---

public final class **ListeImpl**
extends java.lang.Object
implements Liste

Author:

abk640

Constructors

ListeImpl

`public ListeImpl()`

Erezeugt eine neues Listen Objekt ListeImpl.

ListeImpl

`public` **`ListeImpl`**`(java.lang.Object obj)`

Erzeugt eine neue ListeImpl Liste, die leer ist

Parameters:

obj -

ListeImpl

`public` **`ListeImpl`**`(java.util.List list)`

Erstellt ein neue ListeImpl Liste, die mit den Werte aus der Ã¼bergebenen Liste befÃ¼llt wird.

Parameters:

list - = Werte, die in die neue List Ã¼bernommen werden.

Methods

cons

`public void` **`cons`**`(java.lang.Object x)`

Haengt ein Element x an die vorderste Stelle der Liste und markiert dieses als neues Kopfelement.

Parameters:

x - = Anzuhaengender Wert

---

equals

`public boolean` **`equals`**`(java.lang.Object obj)`

StandardmÃ¤ÃŸige equals Methode zum Vergleich von einer List mit einem Any Objekt. Wenn beide denselben Inhalt besitzen, dann wird TRUE zurÃ¼ckgeliefert, ansonsten FALSE.

Parameters:

obj - = Zu vergleichendes ANY Objekt.

Returns:

= Liefert TRUE zurÃ¼ck, wenn der Inhalt beider verlgeichenden Operanden gleich ist, ansonsten FALSE.

Overrides:

equals in class java.lang.Object

---

getKopf

```
public Elem getKopf()
```

Gibt das erste Element, der List zurÃ¼ck.

Returns:

= Erstes Element der Liste

---

head

```
public void head()
```

Entfernt das vorderste Element einer Liste.

---

insert

```
public boolean insert(java.lang.Object x,
                      int n)
```

Fuegt an eine spezifische Stelle in der Liste ein Element an. Dazu iteriert es bis an die Stelle, wo er es einsetzen mÃ¶chte und versetze den Fokus, des Pointer, des vorherigen Elements auf das neue Element. Der Fokus des Pointers, des neuen Elements wird auf das nachfolgende Element gesetzt.

Parameters:

x - = Einzufuegendes Element
n - = Stelle, an der das neue Element eingefuegt werden soll

Returns:

= Gibt TRUE zuruekc, sobald das Einfuegen erfolgreich war

---

isEmpty

```
public boolean isEmpty()
```

Prueft, ob die Liste leer ist.

Returns:

= Gibt TRUE zurueck, wenn sie leer, ansonsten FALSE

---

length

`public int` **`length`**`()`

Gibt die Laenge einer Liste zur. Laenge der Liste wird bestimmt durch die, in der Liste gespeicherten Elemente.

Returns:

= Gibt eine ganzzahlige Laenge einer Liste zurueck

---

referenzenCounter

`public int` **`referenzenCounter`**`()`

Getter- Methode zum Zurueckgeben des Dereferenzierungs- Zaehlers.

Returns:

= Gibt den Dereferenzierungs Zaehler als Integer zurueck.

---

resetReferenzenCounter

`public void` **`resetReferenzenCounter`**`()`

Funktion, zum ZurÃ¼ckstellen des Referenz- Zaehlers.

---

toString

`public java.lang.String` **`toString`**`()`

StandardmaeÃŸige toString Methode, um die Liste in eine nutzbaren String umzukonvertieren.

Returns:

= Gibt die List in Form eines String zurueck.

Overrides:

toString in class java.lang.Object

Class ListeTest

```
java.lang.Object
    |
    +--Uebungsaufgabe_1_2.ListeTest
```

< Constructors > < Methods >

public class **ListeTest**
extends java.lang.Object

Author:
abk640

Constructors

ListeTest

`public` **`ListeTest`**`()`

Methods

testCombined

`public void` **`testCombined`**`()`

Test von allen allen Listenoperationen

---

testCons

`public void` **`testCons`**`()`

---

testConsException

`public void` **`testConsException`**`()`

---

testExceptionHead

`public void` **`testExceptionHead`**`()`

---

testExceptionInsert

`public void` **`testExceptionInsert`**`()`

---

testHead

`public void **testHead**()`

Test of head method, of class Liste.

---

testInsert

`public void **testInsert**()`

Test of insert method, of class Liste.

---

testIsEmpty

`public void **testIsEmpty**()`

Test of isEmpty method, of class Liste.

---

testLength

`public void **testLength**()`

Test of length method, of class Liste.

---

testTask6

`public void **testTask6**()`

Aufgabe 6: Die Funktion cons() liegt in der O-Klasse von O(1). Die neuen Daten werden lediglich am Anfang eingefÃ¼gt. Die anderen Elemente werden nicht beachtet.

---

testTask7

`public void **testTask7**()`

Aufgabe 7: Test zur Aufgabe 7 der Listenimplementation. Um die Elemente an das Ende unserer Liste anzufÃ¼gen, wird dafÃ¼r einfach unsere isnert() Methode verwendet, die als Ziel das EinfÃ¼gen an der letzen Stelle hat. Wir nehmen an, dass die Laufzeit durch das ewige Iterieren an die letze Stelle in O(nÂ²) liegen muss, da fÃ¼r n Elemente n mal Ã¼ber die Liste iteriert werden muss, auch wenn sich die Liste am Anfang noch nicht n groÃŸ ist. Sie baut sich erst mit der Zeit auf.

---

testTask8

```
public void testTask8()
```

Aufgabe 8: Test zu Aufgabe 8 Hypothese: Da wir erneut mit der insert-Methode arbeiten, erwarten wir ein Ã¤hnliches Ergebnis wie bei der Aufgabe 7, allerdings fÃ¼gen wir die Elemente dieses Mal nicht immer an das Ende der Liste an, sondern irgendwo, weshalb der tatsÃ¤chliche Zeitaufwand in Form der Dereferenzierungen etwas niedriger sein sollte, als bei Aufgabe 7 (Stichwort Konstante) Dennoch erwarten wir ein Ergebnis in O(nÂ²). Die Varianz der Laufzeit wird hoch sein, weil wir das Element an einer beliebigen Stelle einfÃ¼gen. Je nachdem wie hoch die Stelle ist, wird die Laufzeit auch verÃ¤ndert.

INDEX