//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ListeADT.java Sist oppdatert 25.02

//

// Definerer først interface til en generell liste. Spesielle typer av

// lister vil bli avledet av dette grensesnittet for å gi et fullt sett

// av nødvendige operasjoner.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.util.Iterator;

**public interface ListeADT<T>**

{

/\*\* Fjerner og returnerer det første elementet fra listen. \*/

**public T fjernFørste ()**;

/\*\* Fjerner og returnerer det siste elementet fra denne listen. \*/

**public T fjernSiste** ();

/\*\* Fjerner og returnerer det spesifiserte elementet fra denne listen.\*/

**public T fjern (T element)**;

/\*\* Returnerer en referanse til det første elementet i denne listen. \*/

**public T første ()**;

/\*\* Returnerer det siste elementet i denne listen. \*/

**public T siste ()**;

/\*\* Returnerer sann hvis denne listen inneholder det spesifiserte

elementet. \*/

**public boolean inneholder (T element)**;

/\*\* Returnerer sann hvis denne listen ikke har elementer. \*/

**public boolean erTom()**;

/\*\* Returnerer antall elementer i listen. \*/

**public int antall()**;

/\*\* Returnerer en strengrepresentasjon av denne listen. \*/

**public String toString();**

}

// TabellListe.java

// Representerer en tabellimplementasjon av en liste. Foran i listen er

// ved indeks 0. Denne Klassen vil bli utvidet ved arv for å gi en   
// spesialisering. Vi sier at en klasse er avledet av en superklasse.

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.util.Iterator;

**class TabellListe<T> implements ListeADT<T>** {

//Det som er deklarert **public** eller **protected** arves.

protected final int STD\_KAPASITET = 100;

protected final int IKKE\_FUNNET = -1;

protected int bak;

protected T[] liste;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oppretter en tom liste med standard kapasitet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public TabellListe()**

{

bak = 0;

liste = (T[])(new Object[STD\_KAPASITET]);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oppretter en tom liste med speisfisert kapasitet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public TabellListe (int startKapsitet)**{

bak = 0;

liste = (T[])(new Object[startKapsitet]);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fjerner og returnerer det siste elementet i denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public T fjernSiste ()**{

T resultat = null;

if (!erTom()){

bak--;

resultat = liste[bak];

liste[bak] = null;

}

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fjerner og returnerer det første elementet i denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public T fjernFørste()**{

T resultat = null;

if (!erTom()){

resultat = liste[0];

bak--;

/\*\* skifer elementene en plass oppover \*/

for (int i=0; i < bak; i++)

liste[i] = liste[i+1];

liste[bak] = null;

}//if

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fjerner og returnerer det spesifiserte elementet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public T fjern** (T element){

T resultat = null;

int indeks = **finn(element)**;

if (indeks != IKKE\_FUNNET){

resultat = liste[indeks];

bak--;

/\*\* skifter elementene etter det vi fjernet en plass opp \*/

for(int i = indeks; i < bak; i++)

liste[i] = liste[i+1];

liste[bak] = null;

}//if

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Returnerer en referanse til første elementet i denne listen.

Elementet blir ikke fjernet.

Hvis listen er tom returnerer null.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public T første()** {

T resultat = null;

if (!erTom())

resultat = liste[0];

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Returner en referanse til elementet bak i denne listen.

Elementet blir ikke fjernet.

Hvis listen er tom returneres null.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public T siste()**{

T resultat = null;

if (!erTom())

resultat = liste[bak-1];

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/Returnerer sann hvis denne listen inneholder det spesifiserte elementet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public boolean inneholder (T el)** {

return (finn(el) != IKKE\_FUNNET);

}

// Returnerer tabellindeksen til det spesifiserte elementet.

// Returnerer IKKE\_FUNNET (-1) hvis ikke funnet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**protected int finn(T el)**{

int i = 0, resultat = IKKE\_FUNNET;

boolean funnet = false;

if (! erTom())

while (! funnet && i < bak)

if (el.equals(liste[i])) //OBS OBS: equals må implementeres i  
 //den klassen som bruker

funnet = true; // metoden, for ellers blir det   
 likhet på referanser.

else // unntatt hvis T er type String

i++;

if (funnet)

resultat = i;

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Returnerer sann hvis denne listen er tom og usann ellers..

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public boolean erTom()**{

return (bak == 0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Returnerer antall elementer i denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public int antall()**{

return bak;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Returnerer en strengrepresentasjon av denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public String toString()**{

String resultat = "";

for (int i=0; i < bak; i++)

resultat = resultat + liste[i].toString() + "\n";

return resultat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oppretter en ny og større tabell for må lagre innholdet av denne  
 listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**protected void utvid()**

{

T[] hjelpeTabell = (T[])(new Object[liste.length\*2]);

for (int i=0; i < liste.length; i++)

hjelpeTabell[i] = liste[i];

liste = hjelpeTabell;

}  
  
}//class

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Ordnet ListeADT.java

// Definerer interface til en ordnet liste.

// Bare elementer som kan sammenlignes blir lagret.

// Nytt element blir lagt til en på riktig plass slik at listen

// er ordnet også etter at det nye elementet er lagt til.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**interface OrdnetListeADT<T > extends ListeADT<T>**

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Legger til det spesifiserte elementet til denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public void leggTil** **(T element)**;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// UordnetListeADT.java

//

// Definerer interface til en uordnet liste. Elementene

// er lagret i en rekkefølge brukeren avgjør.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public interface UordnetListeADT<T> extends ListeADT<T>**

{

/\*\* Legger det spesifiserte elementet foran i denne listen. \*/

**public void leggTilForan**(T element);

/\*\* Legger til det spesifiserte elementet bak i listen. \*/

**public void leggTilBak** (T element);

/\*\* Legger det spesifiserte elementet etter elementet aktuell. \*/

**public void leggTilEtter** (T element, T aktuell);

}  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Nummerert ListeADT.java

// Definerer interface til nummerert liste.

// Elementene blir referert ved numeriske verdier som er påfølgende..

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public interface NummerertListeADT<T> extends ListeADT<T>**{

/\*\* Setter inn det spesifiserte elementet ved det spesifiserte  
 nummer.\*/

**public void leggTil (int nr, T element)**;

/\*\* Setter inn et nytt element ved det spesifiserte nummer.\*/

**public void sett(int nr, T element)**;

/\*\* Legger til det spesifiserte elementet bak i denne listen. \*/

**public void leggTilBak** (T element);

/\*\* Returnerer en referanse til elementet ved den spesifiserte  
 indeksen. \*/

**public T hent(int nr)**;

/\*\* Returnerer nummeret til det spesifiserte elementet. \*/

**public int nrTil**(T element);

/\*\* Fjerner og returnerer elementet ved det spesifiserte nummer. \*/

**public T fjern** **(int nr)**;

}//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// TabellOrdnetListe.java

// Representerer en tabell av en ordnet liste.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Her brukes **interface Comparable** som ligger i java.util.\*

// **public interface Comparable**{

// sammenligner to objekter

// public int compareTo(Object obj);

// Metoden returnerer negativ verdi, 0 eller positiv verdi

// avhengig om this er mindre, lik eller større enn obj

// }

**class TabellOrdnetListe<T > extends TabellListe<T>**

**implements OrdnetListeADT<T>**

**//OBS!leggTil-metoden er effektiv fordi den ikke sammenligner  
// unødvendig mange ganger. Den utnytter at elementene er ordnet.**

**//MEN å bruke fjern-metoden (som arves) til å slette i ordnet liste er  
//ikke effektiv. Mer presis, så er det finn-metoden (som fjern-metoden   
//kaller)som ikke er effektiv. Den utnytter ikke at elementene er ordnet.**

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Opprette en tom liste med standard kapasitet..

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public TabellOrdnetListe()**

{

super();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oppretter en tom liste med spesifisert kapasitet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public TabellOrdnetListe** (int startKapasitet){

super(startKapasitet);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Legger til et spesifisert element til denne listen.

slik at listen fortsatt er sortert.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public void leggTil**(T element)

{

if (antall() == liste.length)

**utvid();** //arves

// **Klassen som skal bruke metoden må implementere Comparable**

**Comparable<T> temp = (Comparable<T>)element;**

int i = 0;

**while (i < bak && temp.compareTo(liste[i]) > 0**)

i++;

// i peker nå på plassen der elementet skal inn

for(int j = bak; j > i; j--)//Må lage plass ved å skifte bakover

liste[j] = liste[j-1]; // de elementene på plassene f.o.m indeks i

liste[i] = element;

bak++;

}

}//class

**class TabellNummerertListe<T>** **extends TabellListe<T>**

**implements NummerertListeADT<T>** {

**public TabellNummerertListe(){**

super();

}

**public TabellNummerertListe(int startKapasitet**){

super(startKapasitet);

}

// ----------------------------------------------------------------

// Ekstra operasjonar for nummerert liste

**public void leggTil(int nr, T element){**// Nytt element inn på gitt nr.   
 //Aukar nr på elementa bak. Bør også kontrollera at verdien nr er gyldig

//

if (bak==liste.length)

**utvid();**

// Flyttar siste del av lista ein plass til høgre

for(int j = bak; j > nr; j--)

liste[j] = liste[j-1];

liste[nr] = element;

bak++;

}

**public void sett(int nr, T element){** // Endrar objektet på plass nr

if(nr >= 0 && nr < bak)

liste[nr] = element;

else

System.out.println("Ugyldig element-nummer");

}

**public void leggTilBak(T element){**

// Legg nytt element inn til slutt i lista

if(bak == liste.length)

**utvid();** //arves

liste[bak] = element;

bak++;

}

**public T hent(int nr**){ // Hentar referanse til objekt på posisjon nr

T svar = null;

if(nr >= 0 && nr < bak)

svar = liste[nr];

return svar;

}

**public T fjern(int nr**){//Fjernar objekt på posisjon nr

T svar = null;

if (nr >=0 && nr < bak){

svar = liste[nr];

for (int i = nr; i<bak-1; i++){

liste[i] = liste[i+1];

}

bak--;

liste[bak] = null;

}

return svar;

}

**public int nrTil(T element){**

// Søkjer posisjonen i den nummererte lista til gitt element

int svar = **finn(element);** // finn arves

return svar;

}

}//class

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TabellUordnetListe.java

// Representerer en tabellimplementasjon av en uordnet liste.

**Class TabellUordnetListe<T> extends TabellListe<T>**

implements UordnetListeADT<T>{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oppretter en tom liste med standard kapasitet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public TabellUordnetListe()**{

super();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oppretter en tom liste med spesifisert kapasitet.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public TabellUordnetListe** (int startKapasitet){

super(startKapasitet);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Legger det spesifiserte elementet foran i denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public void leggTilForan** (T element){

if (antall() == liste.length)

**utvid();**//arves

/\*\* Lager plass ved å skifte elementene bakover \*/

for (int i = bak; i > 0; i--)

liste[i] = liste[i-1];

liste[0] = element;

bak++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Legger det spesifiserte elementet bak I listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public void leggTilBak** (T element){

if (antall() == liste.length)

**utvid();** //arves

liste[bak] = element;

bak++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Legger til det spesifiserte elementet slik at det kommer rett

etter elementet aktuell.

Feilmelding hvis elementet aktuell ikke fins i denne listen.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public void leggTilEtter** (T element, T aktuell){

if (antall() == liste.length)

**utvid();** //arves

int i = 0;

while (i < bak && !aktuell.equals(liste[i]))

i++;

if (i == bak){

System.out.println(" Listen har ikke elementet " + aktuell);

}else{ // Først lage plass og så sette inn

i++;

// i peker nå på plassen etter aktuell, altså der elementet skal inn.

for (int j = bak; j > i; j--) // Skifter de elementene som står

liste[j] = liste[j-1]; // bak element en plass bakover

liste[i] = element;

bak++;

}//if

}

}//class