IN1010 uke 6

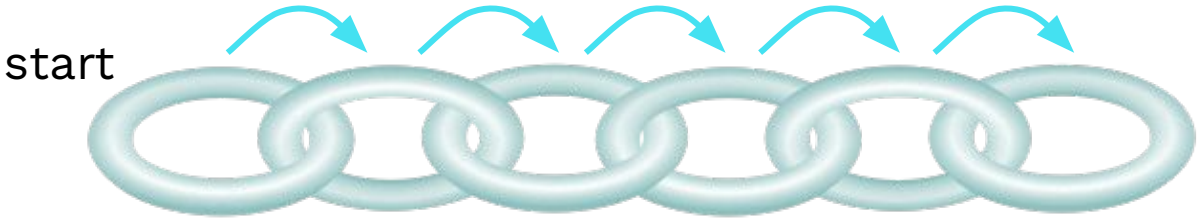
Gruppe 9

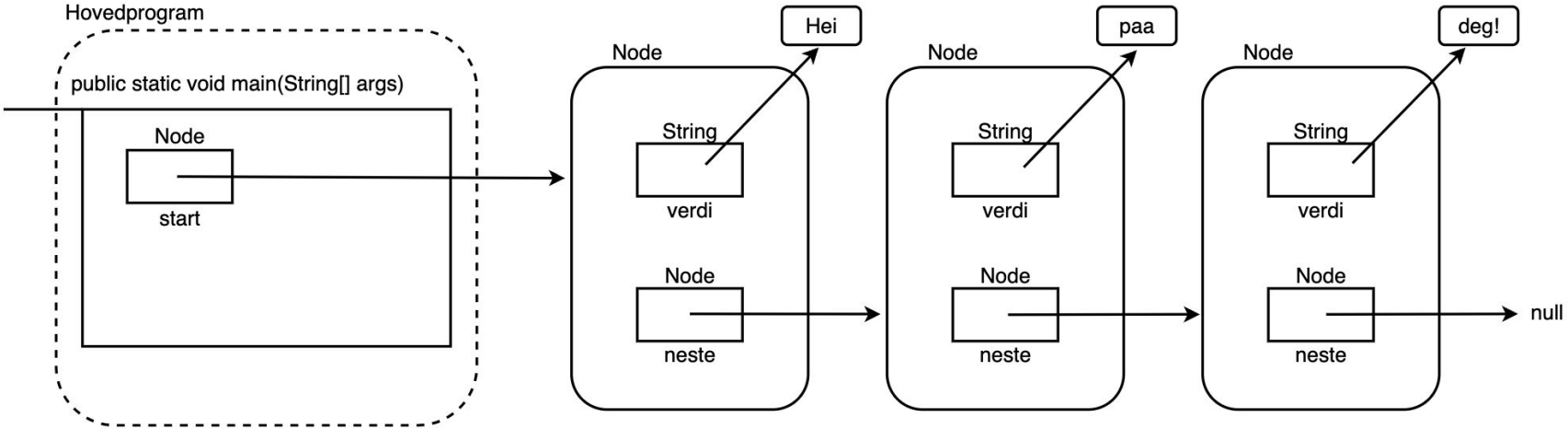
Agenda

- Mer om lister og lenkelister
  - Alternativ: Dobbeltlenket liste
  - Ulike listetyper
    - → FIFO
    - → LIFO
  - Comparable, compareTo
  - Iterable
    - → Kodeeksempel
- Jobbe med oppgaver/trix/oblig!

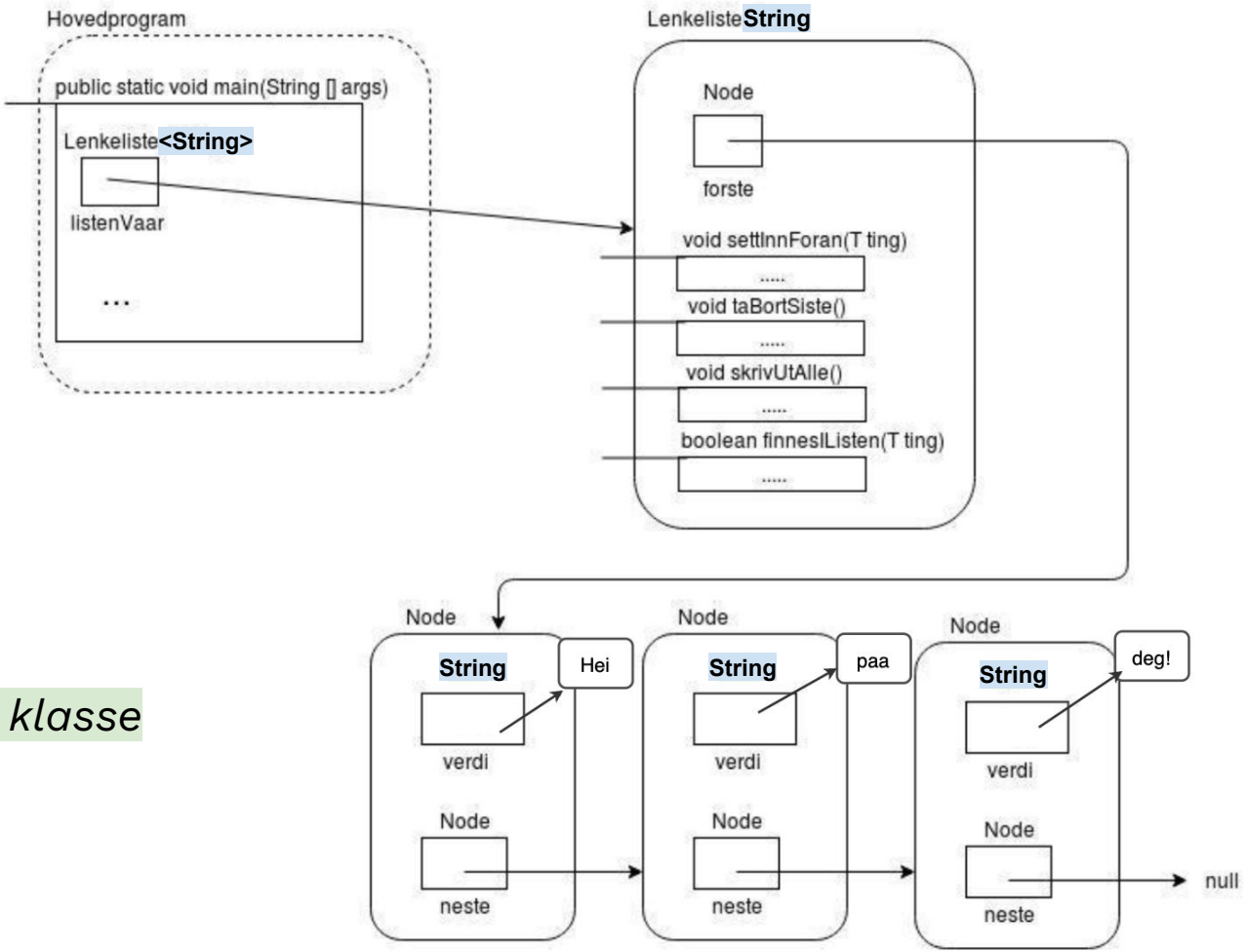
Lenkelister

Lenkeliste - repetisjon

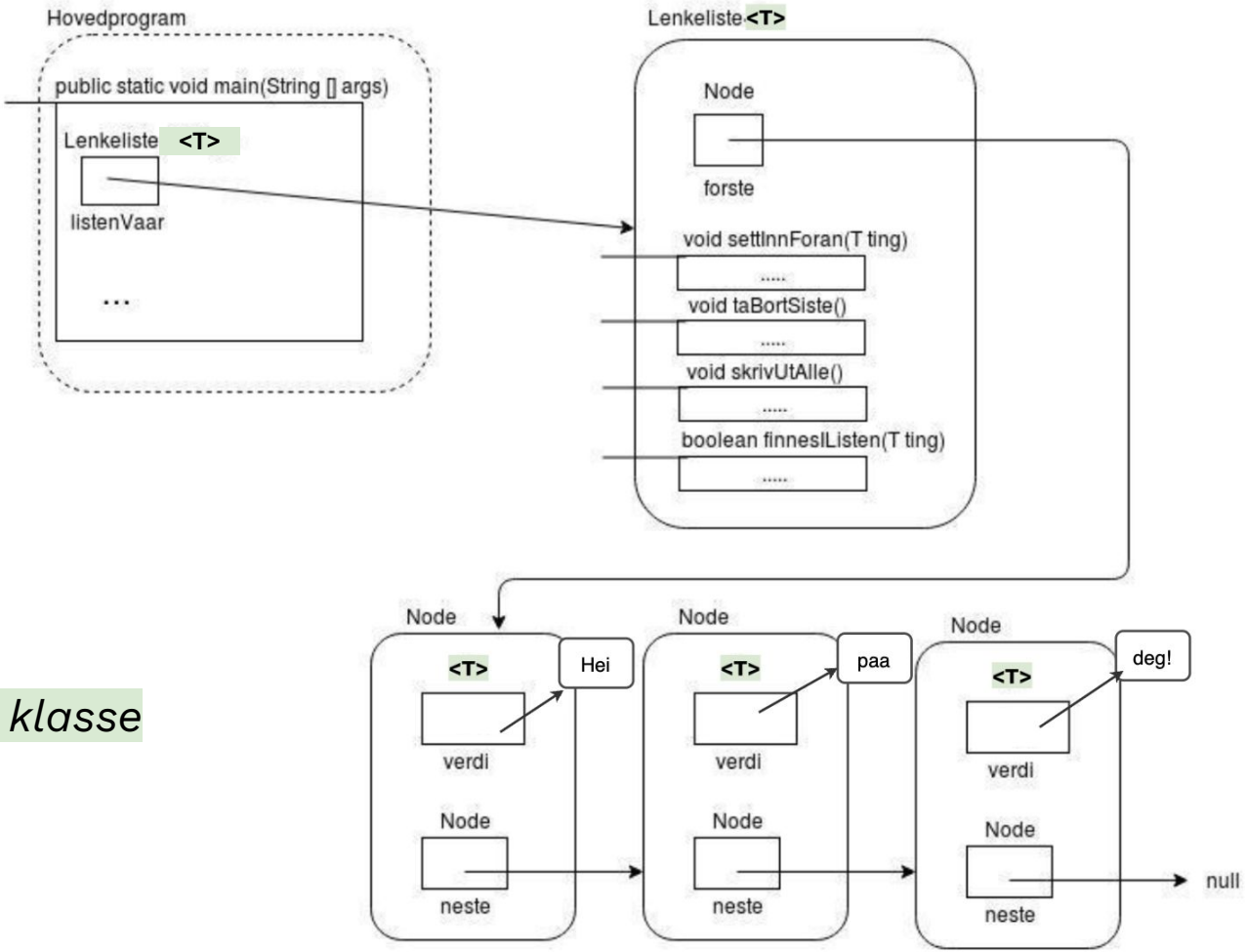




Generisk klasse

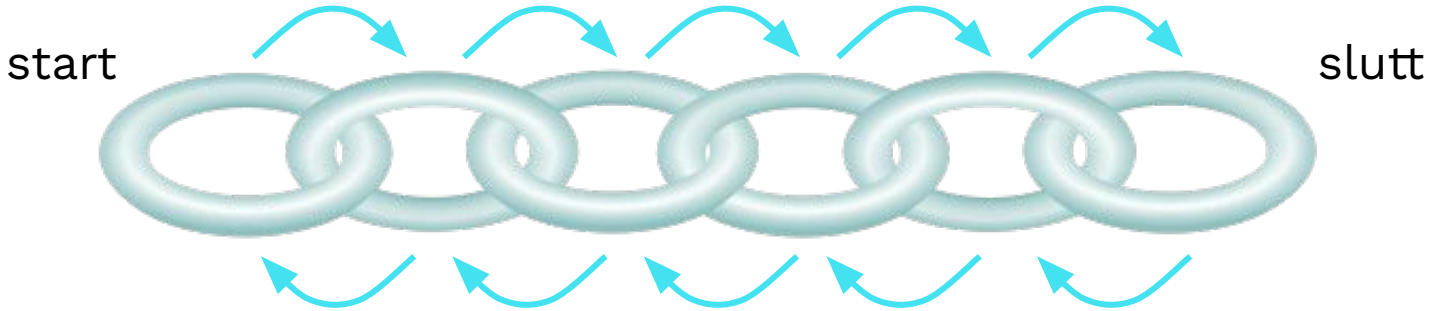


Generisk klasse

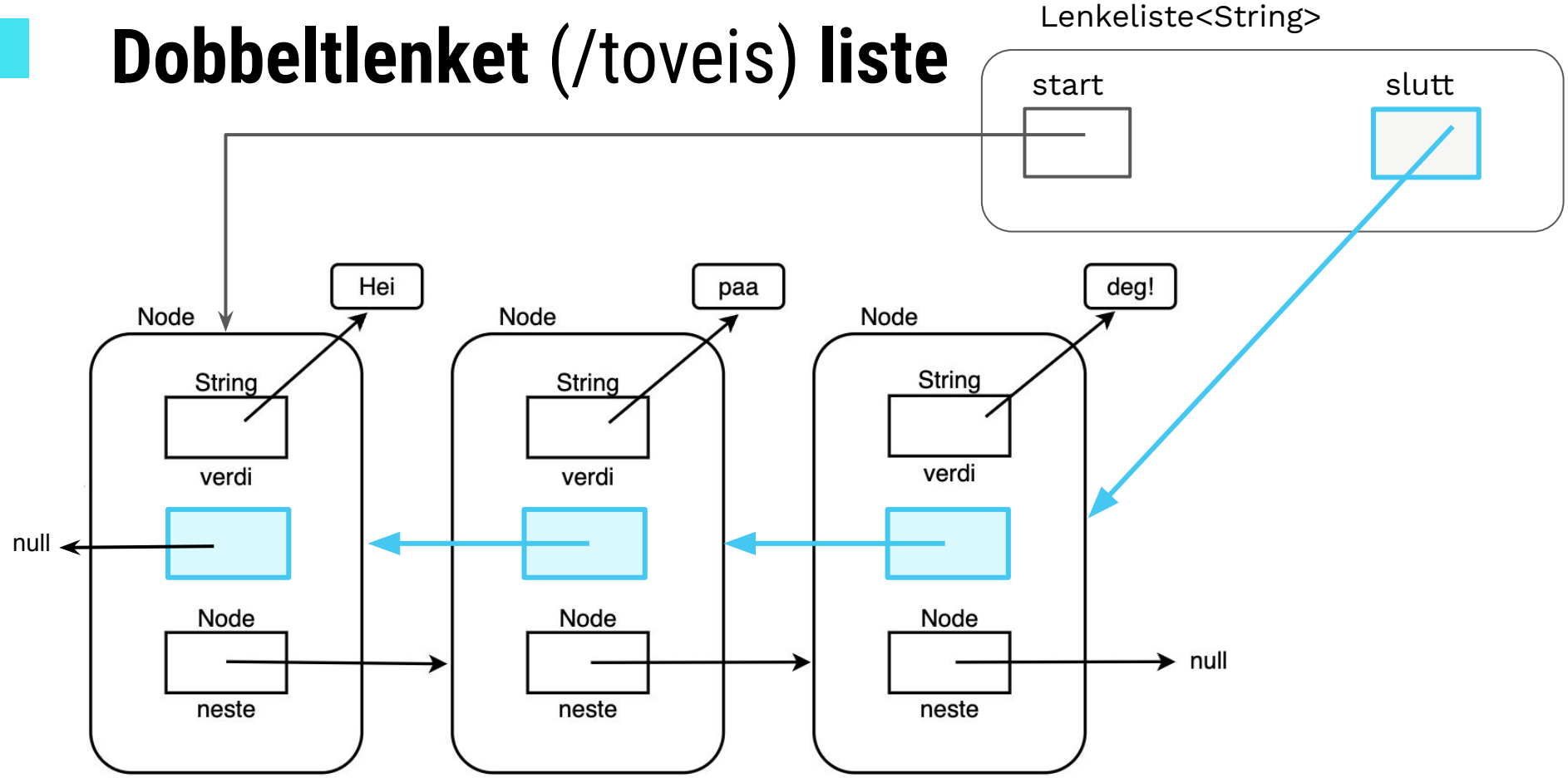


Dobbeltlenket (/toveis) **liste**

+ Kan gå gjennom listen begge retningene
+ Lettere sletting av noder som ikke ligger først i listen
- Mer å holde orden på når man skriver metodene



Dobbeltlenket (/toveis) **liste**



Ulike typer lister

- Kø (FIFO: First in first out)
  - *Sett inn:* Bakerst
  - *Ta ut:* Foran



- Stabel/Stack (FILO: First in last out)
  - *Sett inn:* Foran
  - *Ta ut:* Foran



- Prioritetskø
  - *Ta ut:* Verdi med høyest prioritet

Comparable, compareTo

- *Comparable*: interface med metoden

```
int compareTo(T otherObj);
```

- Implementer hvis vil sammenligne objekter av vår klasse

- ***a**.compareTo(**b**) returnerer en int...*
  - = 0 hvis lik
  - *positivt* tall hvis **a** er *større* enn **b**
  - *negativt* tall hvis **a** er *mindre* enn **b**

compareTo er implementert i *Vare* basert på prisen

Hva burde skrives ut? (0 / positivt tall /negativt tall)?

```
public static void main(String[] args) {
    Vare agurk = new Vare(pris: 20);
    Vare tomat = new Vare(pris: 10);
    Vare melk = new Vare(pris: 20);

    System.out.println(agurk.compareTo(tomat));
    System.out.println(agurk.compareTo(agurk));
    System.out.println(agurk.compareTo(melk));
    System.out.println(tomat.compareTo(melk));
```

compareTo er implementert i *Vare* basert på prisen

Hva burde skrives ut? (0 / positivt tall /negativt tall)?

```
public static void main(String[] args) {
    Vare agurk = new Vare(pris: 20);
    Vare tomat = new Vare(pris: 10);
    Vare melk = new Vare(pris: 20);

    System.out.println(agurk.compareTo(tomat));
    System.out.println(agurk.compareTo(agurk));
    System.out.println(agurk.compareTo(melk));
    System.out.println(tomat.compareTo(melk));
```

Svar:

```
10
0
0
-10
```

Hvordan kan vi implementere *compareTo* i *Vare* basert på pris?

```java
class Vare implements Comparable<Vare> {
    int pris;

    public Vare(int pris) {
        this.pris = pris;
    }

    @Override
    public int compareTo(Vare o) {

    }
```

***a**.compareTo(**b**) returnerer...*

- = 0 hvis lik
- *positivt* tall hvis **a** er *større* enn **b**
- *negativt* tall hvis **a** er *mindre* enn **b**

compareTo er implementert i *Vare* basert på prisen

```
class Vare implements Comparable<Vare> {
    int pris;

    public Vare(int pris) {
        this.pris = pris;
    }

    @Override
    public int compareTo(Vare o) {
        return pris - o.pris;
    }
```

compareTo er implementert i *Vare* basert på prisen

Alternativ:

```java
class Vare implements Comparable<Vare> {
    int pris;

    public Vare(int pris) {
        this.pris = pris;
    }

    @Override
    public int compareTo(Vare o) {
        if (pris > o.pris) return 1;
        if (pris < o.pris) return -1;
        // pris == o.pris
        return 0;
    }
```

Iterable

- *Iterable*: interface med metoden

```
public Iterator<T> iterator()
```

- For å kunne gå gjennom listen med en for-each loop

Iterable

→ Trenger:

Liste<T> implements Iterable<T>

- metoder: iterator()

Indre klasse: ListeIterator implements Iterator<T>

- metoder: hasNext() og next()

Kodeeksempel

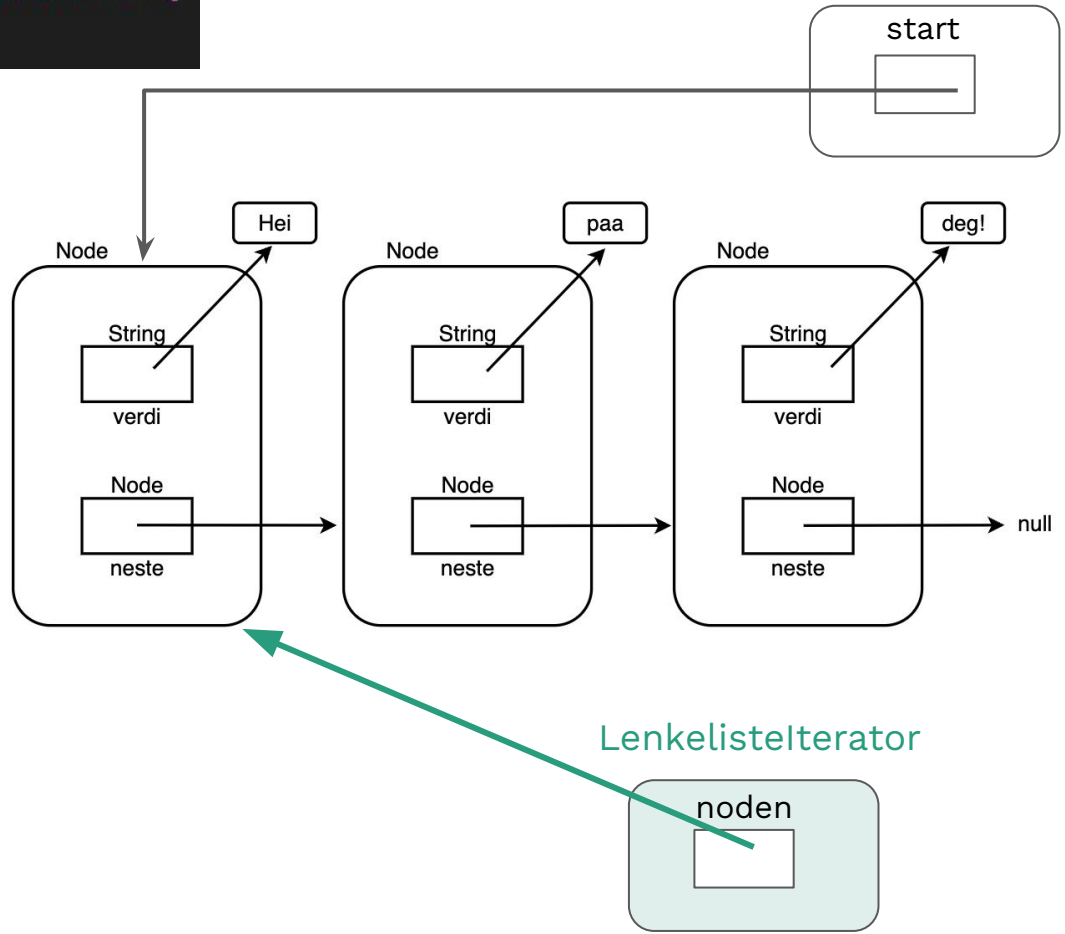
Iterable Lenkeliste

```
private class LenkelisteIterator implements Iterator<E> {
    Node noden = start;
```

```
@Override
public boolean hasNext() {
    return noden != null;
}
```

```
@Override
public E next() {
    if (noden == null) {
        throw new NoSuchElementException(
            s: "next");
    }

    E returVerdi = noden.verdi;
    noden = noden.neste;
    return returVerdi;
}
```

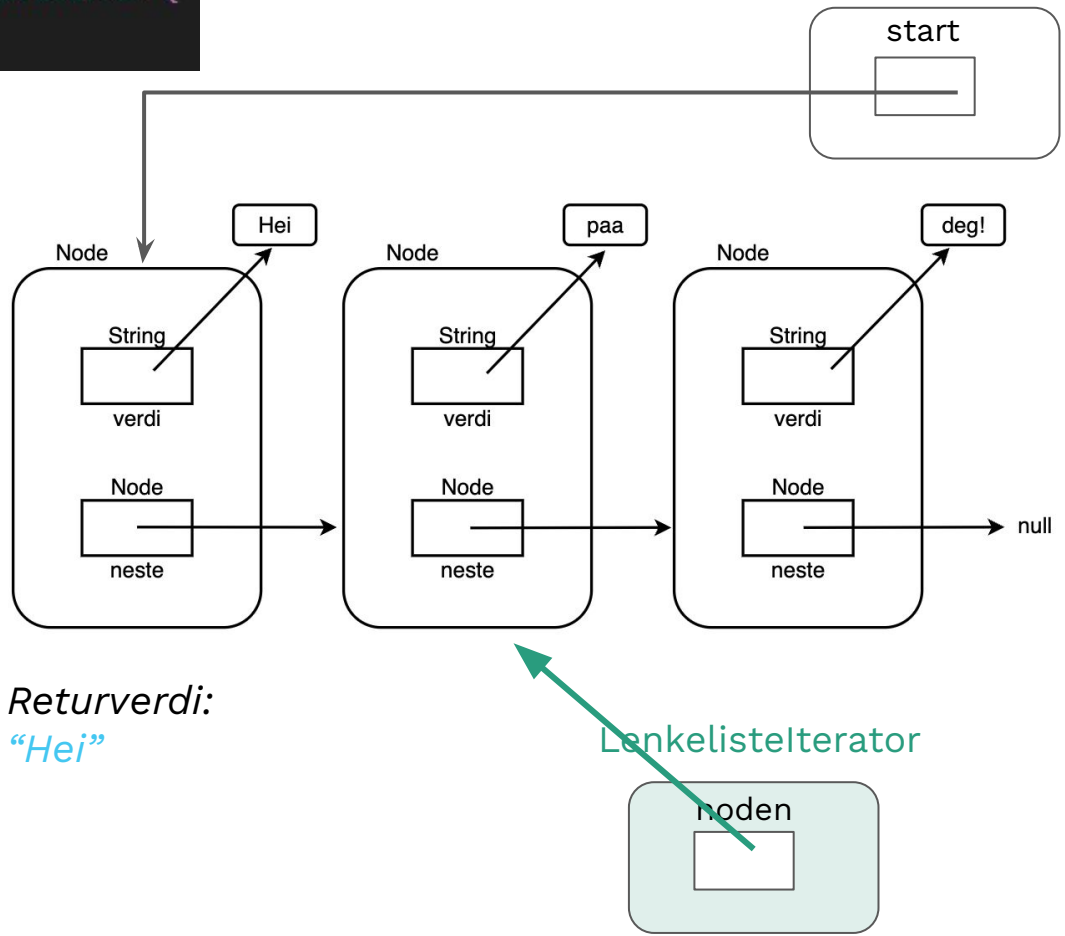


```
private class LenkelisteIterator implements Iterator<E> {
    Node noden = start;
```

```
@Override
public boolean hasNext() {
    return noden != null;
}
```

```
@Override
public E next() {
    if (noden == null) {
        throw new NoSuchElementException(
            s: "next");
    }

    E returVerdi = noden.verdi;
    noden = noden.neste;
    return returVerdi;
}
```

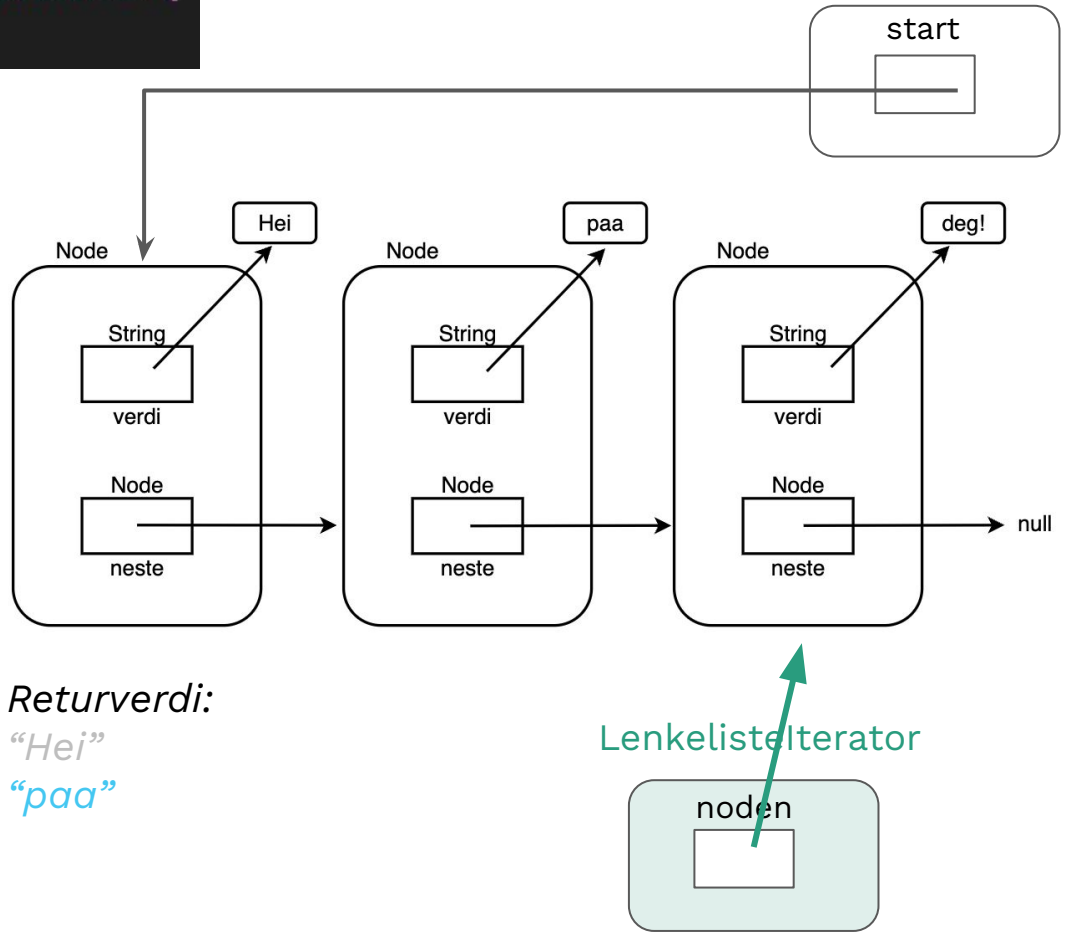


```
private class LenkelisteIterator implements Iterator<E> {
    Node noden = start;
```

```
@Override
public boolean hasNext() {
    return noden != null;
}
```

```
@Override
public E next() {
    if (noden == null) {
        throw new NoSuchElementException(
            s: "next");
    }

    E returVerdi = noden.verdi;
    noden = noden.neste;
    return returVerdi;
}
```

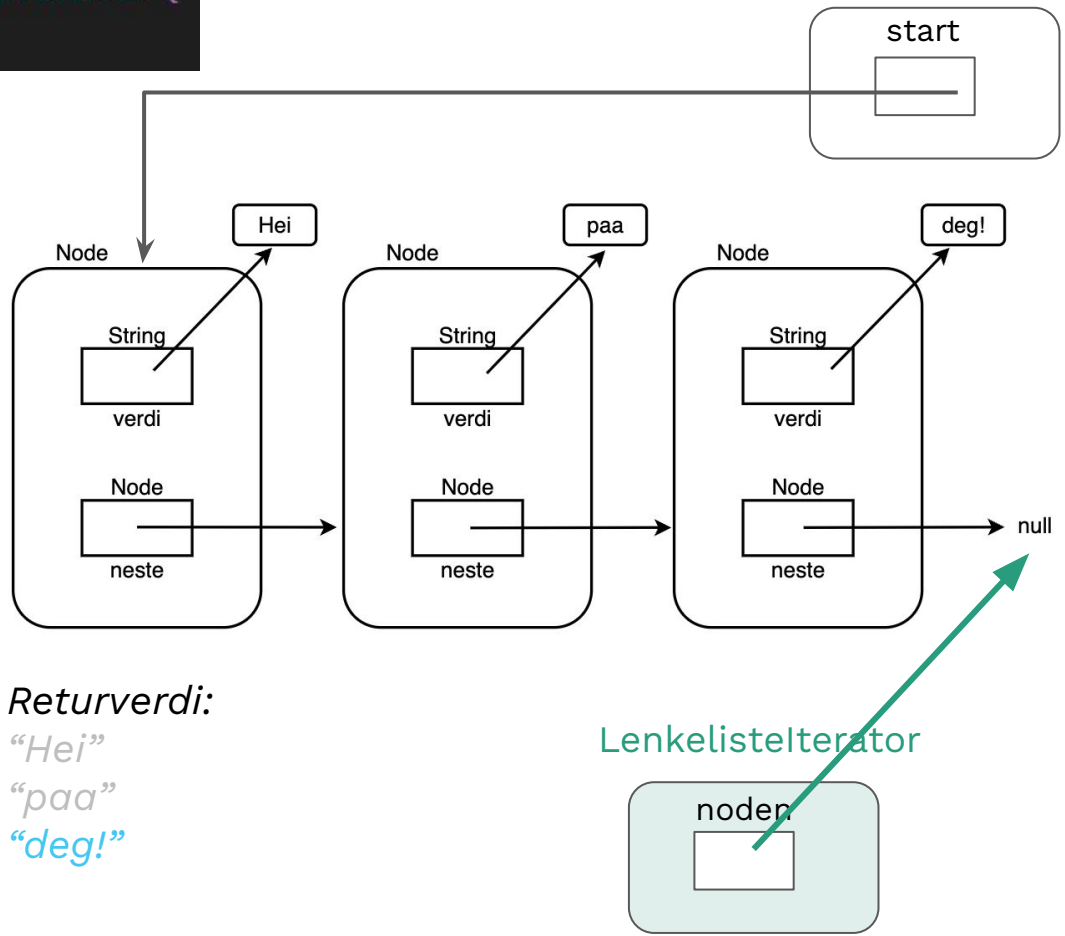


```java
private class LenkelisteIterator implements Iterator<E> {
    Node noden = start;
```

```java
@Override
public boolean hasNext() {
    return noden != null;
}
```

```java
@Override
public E next() {
    if (noden == null) {
        throw new NoSuchElementException(
            s: "next");
    }

    E returVerdi = noden.verdi;
    noden = noden.neste;
    return returVerdi;
}
```



Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 9 → Uke6