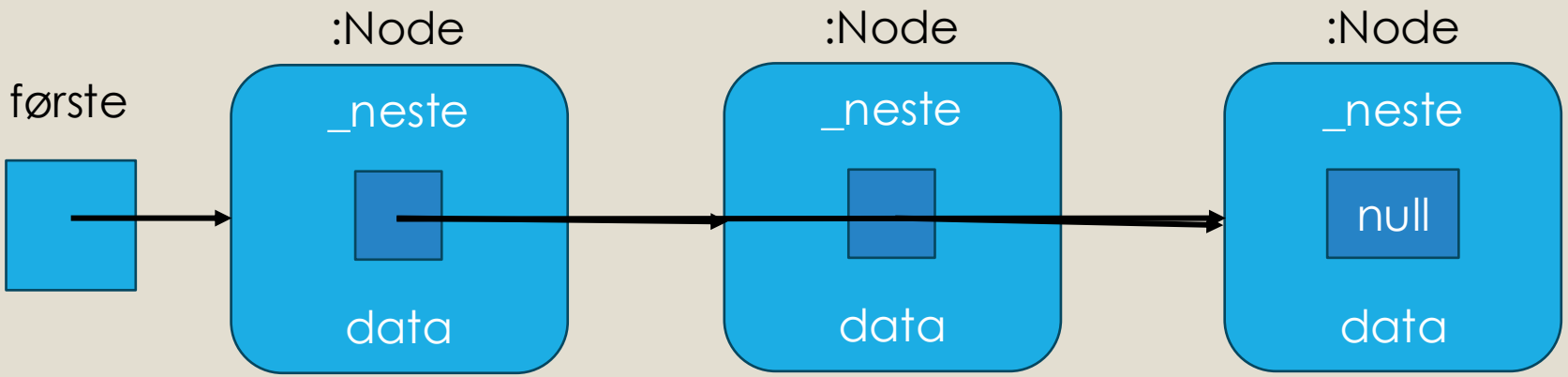
GRUPPE 1

Velkommen til
IN1010 – Repetisjon

Plan for dagen

- Lenkede lister
  - Enkelt
  - Dobbelt
- Exceptions
- Generiske typer
- Datastrukturer
  - Stabel, kø og prioritetskø
- Iterable
- Comparable

Lenkede lister



Exceptions

```java
public class UlovligListeIndeks extends RuntimeException {
    public UlovligListeIndeks(int pos, int max) {
        super("Listeindeks " + pos + " ikke i intervallet 0-" + max);
    }
}
```

```java
public void remove(int pos) {
    if (pos < 0 || pos >= antallNoder) {
        throw new UlovligListeIndeks(pos, antallNoder - 1);
    }
    if (pos == 0) {
        første = første.neste;
    } else {
        Node iter = første;
        for (int i = 0; i < pos - 1; i++) {
            iter = iter.neste;
        }
        iter.neste = iter.neste.neste;
    }
    antallNoder--;
}
```

```java
public class GeneriskLenketliste<T> {

    private class Node {
        T innhold;
        Node neste;

        public Node(T innhold, Node neste) {
            this.innhold = innhold;
            this.neste = neste;
        }
    }

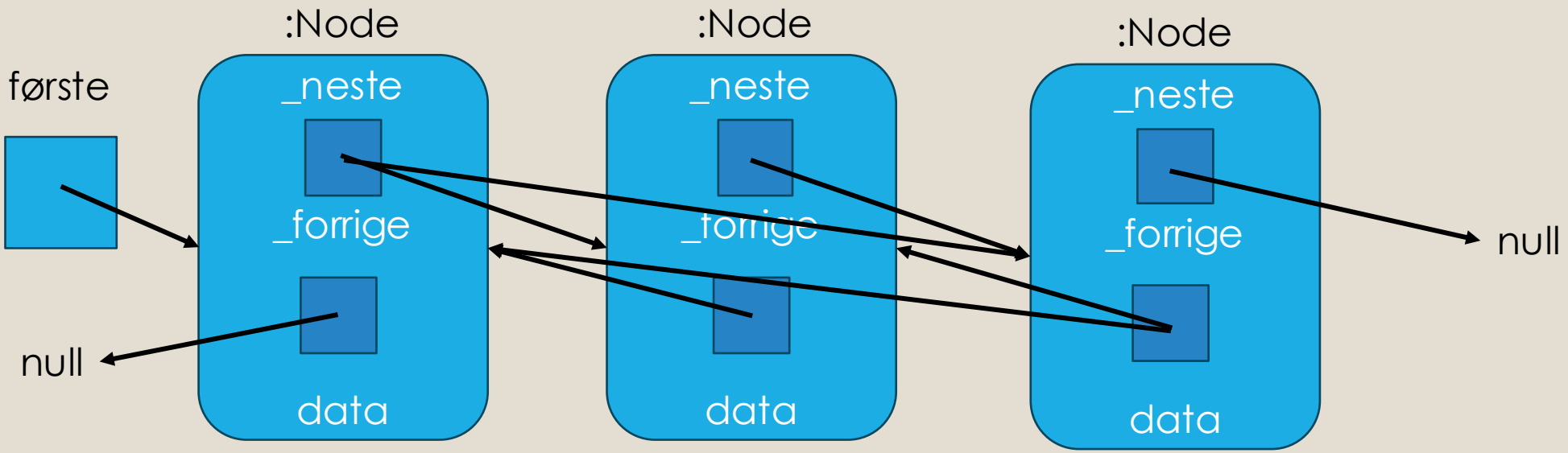
    private Node første;
    int antallNoder = 0;

    public void settInnFørst(T innhold) {
        Node ny = new Node(innhold, første);
        første = ny;
        antallNoder++;
    }
```

Generiske typer

- Lar oss lage beholdere som fungerer med alle typer
- De fleste beholdere i Java er generiske

Dobbelt lenket liste



Dobbelt lenket liste

```java
private class Node {
    String innhold;
    Node neste;
    Node forrige;

    public Node(String innhold, Node neste, Node forrige) {
        this.innhold = innhold;
        this.neste = neste;
        this.forrige = forrige; // legg til forrige peker
    }
}
```

```java
public void settInnFørste(String innhold) {
    Node ny = new Node(innhold, første, null);

    if (første != null) {
        første.forrige = ny; // sette peker tilbake nye første
    }

    første = ny;
    antallNoder++;
}
```

```java
public void settInnSiste(String innhold) {
    Node ny = new Node(innhold, null, null);

    if (første == null) {
        første = ny;
        antallNoder++;
        return;
    }

    Node itNode = første;

    while (itNode.neste != null) {
        itNode = itNode.neste;
```

```java
public Node fjern(int pos) {
    Node itNode = første;

    if (pos < 0 || pos >= antallNoder) {
        throw new UlovligListeIndeks(pos, antallNoder - 1);
    }

    for (int i = 0; i < pos; i++) {
        itNode = itNode.neste;
    }

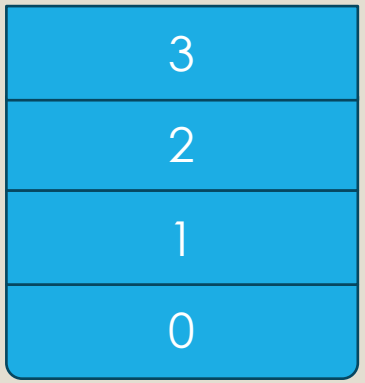
    if (itNode.forrige != null) {
        itNode.forrige.neste = itNode.neste;
    } else {
        første = itNode.neste;
    }

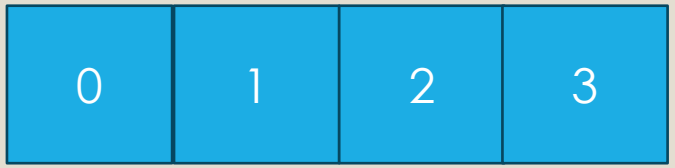
    if (itNode.neste != null) {
        itNode.neste.forrige = itNode.forrige;
    }

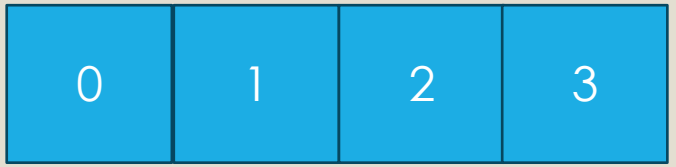
    antallNoder--;
    return itNode;
}
```

Datastrukturer

- Stack
- Kø
- Prioritets kø







Iterable

```java
for (int i = 0; i < liste.length; i++) {
    System.out.println(liste[i]);
}
```

```java
for (int tall : liste) {
    System.out.println(tall);
}
```

```java
public interface Iterator<T> {
    boolean hasNext();

    T next();
}
```

- Lar oss iterere gjennom elementene i en beholder ett for ett
- Holder orden på hvor langt vi har kommet gjennom og sjekker om det finnes flere elementer

```java
public class Liste<T> implements Iterable<T> {
    private class Node {
        T innhold;
        Node neste;

        public Node(T innhold, Node neste) {
            this.innhold = innhold;
            this.neste = neste;
        }
    }

    private Node første;

    @Override
    public Iterator<T> iterator() {
        return new ListeIterator();
    }

    class ListeIterator implements Iterator<T> {
        private Node denne;

        public ListeIterator() {
            denne = første;
        }

        @Override
        public T next() {
            T midlertidig = denne.innhold;
            denne = denne.neste;
            return midlertidig;
        }

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

Iterable

```java
import java.util.Iterator;
```

```java
public static void main(String[] args) {
    Liste<Integer> liste = new Liste<>();

    liste.settInn(1);
    liste.settInn(2);
    liste.settInn(3);

    for (int tall : liste) {
        System.out.println(tall);
    }
}
```

```java
public class Katt implements Comparable<Katt> {
    String navn;
    int alder;

    public Katt(String n, int a) {
        navn = n;
        alder = a;
    }

    public int compareTo(Katt katt) {
        if (alder > katt.alder) {
            return 1;
        }

        else if (alder < katt.alder) {
            return -1;
        }

        return 0;
    }
}
```

Comparable

- Lar oss definere hvordan vi sammenlikner egne objekter

```java
public static void main(String[] args) {
    Katt nala = new Katt("Nala", 13);
    Katt kiara = new Katt("Kiara", 14);

    if (nala.compareTo(kiara) == 1) {
        System.out.println("Nala er eldst");
    } else if (nala.compareTo(kiara) == -1) {
        System.out.println("Kiara er eldst");
    } else {
        System.out.println("De er like gamle");
    }
}
```