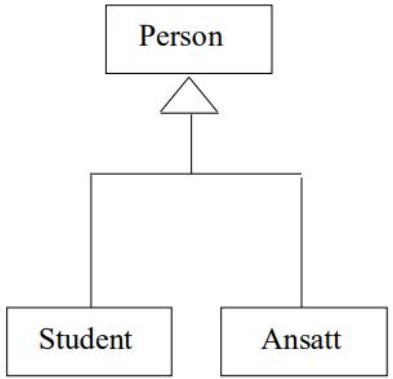
IN1010 - Gruppe 11

Uke 5

Plan for dagen

- Array og ArrayList
- Generiske klasser
- Lenkelister

Når du planlegger programmet



Når du skriver programmet

```java
class Person {
    String navn;
    int tlfnr;
    boolean gyldigTlfnr() { ... }
}
```

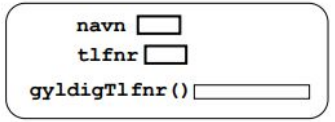
```java
class Student extends Person {
    String program;

    void byttProgram(String nytt) {
        program = nytt;
    }
}
```

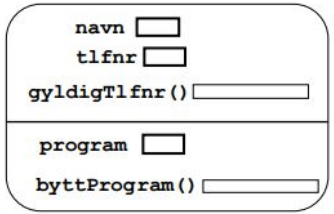
```java
class Ansatt extends Person {
    int lønnstrinn;
    float antallTimer;

    void lønnstillegg(int tillegg) {
        lønnstrinn += tillegg;
    }
}
```

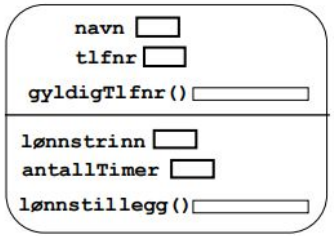
Når du kjører programmet



Person-objekt



Student-objekt



Ansatt-objekt

ArrayList vs. Array (overblikk)

- **ArrayList**
  - Dynamisk størrelse
  - Må inneholder klasser
    - Feks Integer (og ikke int)
  - Mange innebygde metoder
    - Oversikt over metodene finnes i dokumentasjonen :)

```
//opprette ArrayList:
ArrayList<Integer> tall1 = new ArrayList<>();

//legge til element:
tall1.add(e:1);

//hente element på en indeks:
int i = tall1.get(index:0);

//størrelse:
int str1 = tall1.size();
```

- **Array**
  - Fast størrelse
  - Ikke like mange metoder
  - Kan også inneholder primitive typer
    - feks int men også integer

```
//opprette Array:
int[] tall2 = new int[5];

//legge til element
tall2[0] = 1;

//hente element på indeks:
int j = tall2[0];

//størrelse
int str2 = tall2.length;
//str2 == 5
```

Generiske klasser

- **Ønsker ofte at beholdere skal fungere til ulike typer/klasser**
  - Tungvint å lage en type beholder for integers og en annen type for String
- Beholderen kan ha parameter i klasse deklarasjonen
  - class Beholder { … }
- Bestemmer typen/klassen beholderen skal ha i det man oppretter en instans av beholderen

```
ArrayList<Integer> tall = new ArrayList<>();
ArrayList<String> strings = new ArrayList<>();
```



Generiske klasser

- **T** er en placeholder for en type.
- Må spesifisere typen **T** i det vi oppretter en instans av Boks

- **E** - Element
- **K** - Key
- **N** - Number
- **T** - Type
- **V** - Value

```java
public class Boks<T> {
    private T t;

    public void fyll(T t) {
        this.t = t;
    }

    public T hent() {
        return t;
    }
}
```

```java
Run | Debug
public static void main(String[] args) {

    Boks<String> b = new Boks<String>();
    String tekst = "hallo!";

    b.fyll(tekst);
    System.out.println(b.hent()); //"hallo!"

}
```

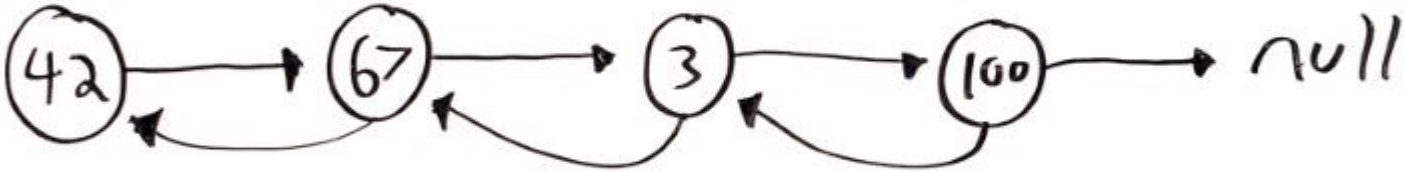
Lenkelister

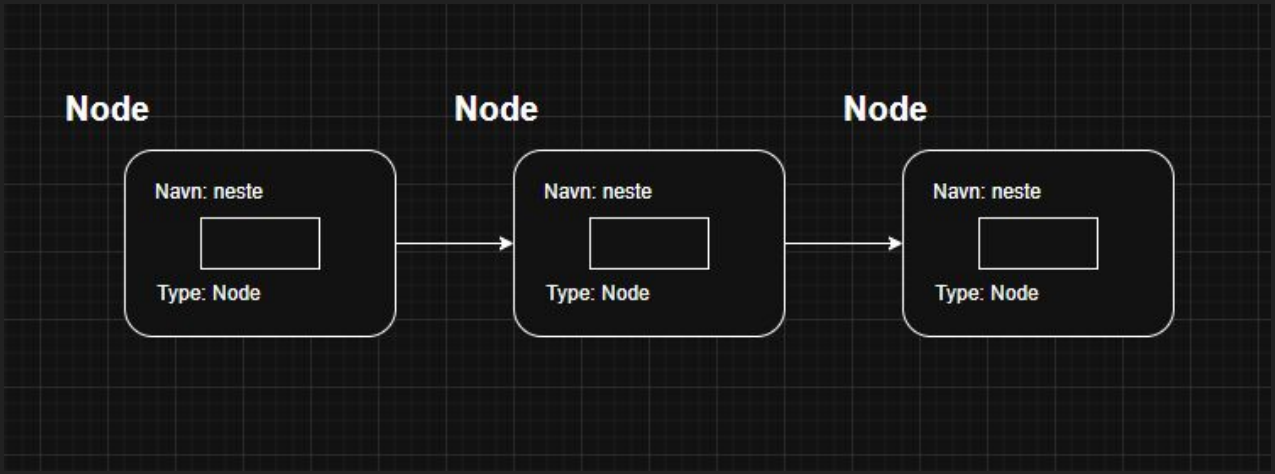
- Kan implementere egne beholdere.
- En hjelpeklasse Node.
  - Kan hete noe annet, men det er vanlig å kalle den Node.
- Node har en referanse til neste node i lenkelisten.
- Alle noder har dermed en referanse til neste node
  - **Unntak**: siste node

```
class Node {
    Node neste; //= null hvis ingen neste

    public void settNeste(Node n){
        neste = n;
    }

    public Node hentNeste(){
        return neste;
    }
```


42
67
3
100
null




```
class Node {
    Node neste; //= null hvis ingen neste

    public void settNeste(Node n){
        neste = n;
    }

    public Node hentNeste(){
        return neste;
    }
```

Lenkelister

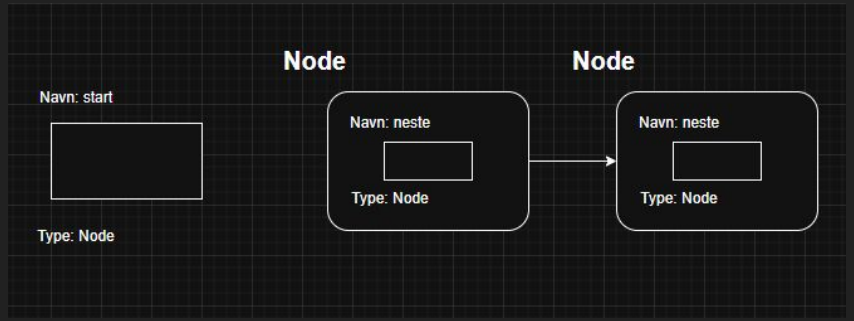
- Må ha en referanse til første noden i lenkelisten for å kunne nå alle elementene i lenkelisten.

- Starter med å opprette en startnode (start)
- Oppretter en node som skal komme etter startnoden (nrTo)
- Setter nrTo til å være start sin neste

```
class Node {
    Node neste; //= null hvis ingen neste

    public void settNeste(Node n){
        neste = n;
    }

    public Node hentNeste(){
        return neste;
    }
```



Lenkelister

- Node-klassen vi har gått gjennom hittil er ganske ubrukelig da den ikke holder på noe data
  - Kun en peker til neste
- Ofte ønsker vi at en Node inneholder:
  - En peker til neste node (eller null)
  - Data

```
class Node {
    Node neste; //= null hvis ingen neste
    String verdi;

    //Konstruktør:
public Node(String verdi){
        this.verdi = verdi;
    }

    public void settNeste(Node n){
        neste = n;
    }

    public Node hentNeste(){
        return neste;
    }

    public String hentData(){
        return verdi;
    }
}
```

Lenkelister

“Demo”

Ulike typer lister / lenkelister

- **Kø / queue**– first in first out (**FIFO**)
  - Som en vanlig kø i butikken.
- **Stabel/stack** – last in first out (**LIFO**)
  - Som tallerkener som man stabler oppå hverandre.
- **Prioritetskø** – Den med høyest prioritet først
  - Feks den sykeste pasienten på legevakten

```
class Node {
    Node neste; //= null hvis ingen neste
    String verdi;

    //Konstruktør:
public Node(String verdi){
        this.verdi = verdi;
    }

    public void settNeste(Node n){
        neste = n;
    }

    public Node hentNeste(){
        return neste;
    }

    public String hentData(){
        return verdi;
    }
}
```

Lab!