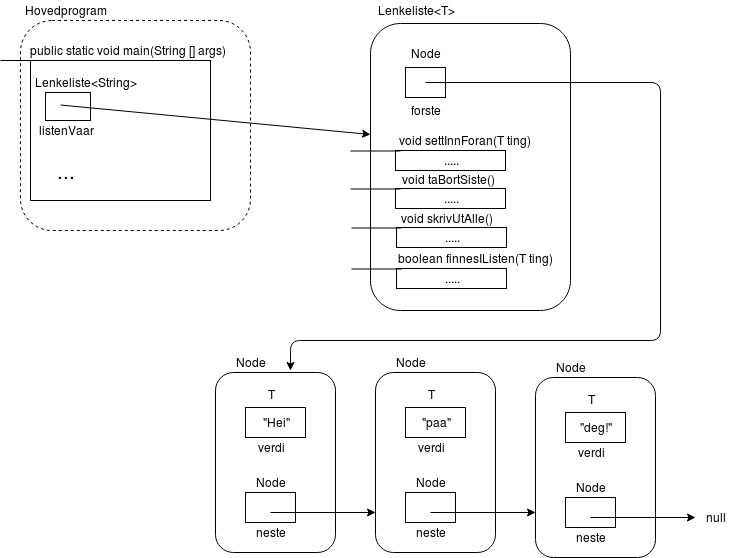
## Seminaropplegg uke 7

* Fortsettelse av generiske klasser og lenkelister.
* Comparable, compareTo.
  + class Klasse implements Comparable<KlassenViVilSammenlign eMed>
* Introdusere klassen “Lenkeliste”. Kan også være generisk.
* Gi eksempler på exceptions: <https://trix.ifi.uio.no/assignments/557>

Tegn/vis eksempel, lenkelisten vil da håndtere all innsetting, sletting og andre operasjoner vi ønsker å gjøre på listen. Vi trenger derfor ikke referanser til nodene i main, men har en referanse til lenkelisten og kan gjennom denne få tilgang til dataen lagret i nodene våre.

(Her med noen eksempler på metoder i Lenkelister, kan ha mange fler eller færre enn de som er her).



* FIFO/LIFO. Tegn gjerne opp forskjellen.
  + FIFO-liste - First in, first out. Ex. Kø. Setter inn nye elementer bakerst.
  + LIFO-lista - Last in, first out. Ex. Stack. Setter inn nye elementer forrerst.
* Indre klasser

**Iterator-delen er gjerne tenkt til å være bare live-koding i slutten av timen. Det forventes ikke at de skal gjøre dette selv.**

**Oppgaver kan finnes her:** [**https://github.uio.no/IN1010/seminaroppgaver\_v19/blob/master/uke07.md**](https://github.uio.no/IN1010/seminaroppgaver_v19/blob/master/uke07.md)

**Oppgave 1**

Implementer klassen “Person”. En person skal ha ett fornavn, ett etternavn og en alder, disse skal settes i konstruktøren.

**1a.**

Implementer en toString-metode.

**1b.**

Implementer en metode som henter alder.

**1c.**

Implementer en metode “int compareTo(Person annen)”. Når vi sammenligner personer ønsker vi først å fremst å sortere på alder. Dersom alder er lik så ønsker vi å sortere på navn. *Hint: her kan du gjøre kall på både hentAlder() og toString().*

**1d.**

Test programmet ditt med denne kodesnutten…

|  |
| --- |
| import java.util.Arrays;  class Testprogram {  public static void main(String[] args) {  Person[] personer = {  new Person("Andreas", "Hurthe", 25),  new Person("Kristine", "Henriksen", 30),  new Person("Henrik", "Golsen", 25),  new Person("Sigvart", "Dagsland", 29)  };  Arrays.sort(personer);  for (Person p : personer) {  System.out.println(p);  }  }  } |

**Løsning:**

// Comparable Person  
class Person implements Comparable<Person> {  
 String fornavn;  
 String etternavn;  
 int alder;  
  
 public Person(String f, String e, int a){  
 fornavn = f;  
 etternavn = e;  
 alder = a;  
 }  
   
 public int compareTo(Person annen){  
 if(alder == annen.getAlder()){  
 return this.toString().compareTo(annen.toString());  
 }  
 return alder - annen.getAlder();  
 }  
  
 public String toString(){  
 return etternavn + ", " + fornavn + ": " + alder;  
 }  
  
 public int getAlder(){  
 return alder;  
 }  
}

**Oppgave 2**

Lag en generisk klasse “Lenkeliste”. Dere kan ta utgangspunkt i grensesnittet gitt under. Dere kan benytte dere av Nodeklassen skrevet forrige uke, men den skal nå flyttes inn i klassen Lenkeliste som en indre/privat klasse. *Diskusjon: Hva slags endringer må gjøres på Node-klassen?*

|  |
| --- |
| public class FIFOListe<T> {  private Node hode;  private Node hale;  private class Node {}  public int storrelse() {}  public boolean erTom() {}  public void leggTil(T t) {}  public void fjernAlt() {}  public Node hentNode(int indeks) {}  public T hent(int indeks) {}  public T fjern(int indeks) {}  public void skrivUtListe() {}  } |

Løsning:

|  |
| --- |
| class FIFOListe<T> implements Iterable<T> {  private Node hode;  private Node hale;  private class Node {  Node neste;  T ting;  Node(T ting) {  this.ting = ting;  }  }  public int storrelse() {  int teller = 0;  for (Node tmp = hode; tmp != null; tmp = tmp.neste) {  teller++;  }  return teller;  }  public boolean erTom() {  return hode == null;  }  public void leggTil(T t) {  Node nyNode = new Node(t);  if (erTom()) {  hode = nyNode;  }  else {  hale.neste = nyNode;  }  hale = nyNode;  }  public void fjernAlt() {  hode = null;  hale = null;  }  public T pop() {  Node returnode = hode;  hode = hode.neste;  return returnode.ting;  }  public void skrivUtListe() {  Node tmp = hode;  System.out.println("[");  while (tmp != null) {  System.out.println("\t" + tmp.ting);  tmp = tmp.neste;  }  System.out.println("]");  }  public Iterator<T> iterator() {  return new FIFOIterator();  }  private class FIFOIterator implements Iterator<T> {  Node denne = hode;  @Override  public boolean hasNext() {  return denne != null;  }  @Override  public T next() {  if (denne == null) throw new NoSuchElementException("next")  Node tmp = denne;  denne = denne.neste;  return tmp.ting;  }  }  } |

|  |
| --- |
| class TestFIFO {  public static void main(String[] args) {  FIFOListe<String> liste = new FIFOListe<>();  liste.leggTil("Hei");  liste.leggTil("på");  liste.leggTil("deg");  liste.leggTil("!");  System.out.println("\nBruker skrivUtListe() ... ");  liste.skrivUtListe();  System.out.println("\nBruker iterator ... ");  for (String string : liste) {  System.out.print(string + " ");  }  System.out.println();  System.out.println("\nBruker pop() ... ");  System.out.print(liste.pop() + " ");  System.out.print(liste.pop() + " ");  System.out.print(liste.pop());  System.out.println(liste.pop());  }  } |