Seminar løsning

Oppgave 1

```
import java.util.Iterator;

class VaarArray<T> implements Iterable<T> {
  T [] arr;

  public VaarArray(int strl) {
         arr = (T[]) new Object[strl];
  }

  //Finner ledig plass og setter inn. Hvis det er plass returneres true,
  //hvis det er fullt returneres false
  public boolean settInn(T elem){
       for(int i = 0; i < arr.length; i++){
         if (arr[i] == null) {
            arr[i] = elem;
            return true;
         }
       }
       return false;
  }

  //Returnerer true dersom det var mulig å fjerne, false ellers.
  public boolean fjernFraIndex(int n){
    //Hvis ugyldig index
       if(arr.length <= n || n < 0){
               return false;
       }
    //Hvis plassen er tom
       if (arr[n] == null){
               return false;
       }
       arr[n] = null;
       return true;
  }


  public Iterator<T> iterator(){
         return new ArrayIterator();
  }


  private class ArrayIterator implements Iterator<T>{
    int pos = 0;

    public boolean hasNext(){
```

```
            for(int i = pos; i < arr.length; i++){
               if(arr[i] != null) {
                        return true;
               }
            }
            return false;
    }

    public T next(){
         for(int i = pos; i < arr.length; i++){
           if(arr[i] != null) {
               T returverdi = arr[i];
               pos ++;
               return returverdi;
            }
          }
          return null;
      }
    }

}
```

Oppgave 2

2a)

```
// Forklar hvorfor det er brukt overloading, mao.
// Fordi vi vet at vi alltid vil begynne med hode-noden,
// men da trenger man ikke fra hovedprogrammet å vite dette.
  public void skriv_rek_forst(){
    skriv_rek_forst(hode);
  }
  public void skriv_rek_forst(Node denne){
    if (denne == null){
        return;
    }
    System.out.println(denne.data);
    skriv_rek_forst(denne.neste);
  }

  public void skriv_rek_sist(){
    skriv_rek_sist(hode);
  }
  public void skriv_rek_sist(Node denne){
    if (denne == null){
        return;
    }
    skriv_rek_sist(denne.neste);
```

```
    System.out.println(denne.data);
  }
```

2b)

```
public int antNoder(){
    return antNoder(hode, 0);
  }

  public int antNoder(Node denne, int ant){
    if (denne == null){
        return ant;
    }
    return antNoder(denne.neste, ant+1);
  }
```