Seminar oppgaver

Iterator

→ Hvorfor? Kan løkke oss gjennom elementen i klassen vi har laget

- Klassen vi ønsker å iterere over: `implements Iterable<T>`
- Grensesnittet Iterable består av 1 metode vi må implementere: `public Iterator<T> iterator();`
- Iteratorklassen: `implements Iterator<T>`
- Grensesnittet Iterable består av 2 metoder vi må implementere (som vi må forholde oss til i dette kurset):
  - `public T next();`
  - `public boolean hasNext();`

Oppgave 1

1a)

Gitt følgende grensesnitt, implementer koden til “settInn”. Den finner en ledig plass i arrayet og setter inn elementet, hvis det går fint (og det er en ledig plass) så returneres true, hvis det er fullt returneres false.

1b)

Implementer “fjernFraIndex(int n)”. Den fjerner et element fra en gyldig indeks. Dersom indeks n ikke finnes eller plassen allerede er tom returneres false, ellers settes plassen til null og true returneres.

```
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);

  //Returnerer true dersom det var mulig å fjerne, false ellers.
  public boolean fjernFraIndex(int n);

  public Iterator<T> iterator();

  private class ArrayIterator implements Iterator<T>{
  //variabler..?

      public boolean hasNext();
      public T next();
  }
}
```

1c)

Implementer iterator()-metoden, samt den indre klassen ArrayIterator. Den må ha en variable som holder styr på hvor langt vi har kommet i itereringen. hasNext metoden skal sjekke om det finnes flere elementer i listen, returnerer true hvis det finnes flere, false ellers. next() henter ut det neste elementet i listen, og oppdaterer da variablen som holder styr på hvor langt vi har kommet i itereringen.

Rekursjon

Enkelt forklart: en metode som kaller på seg selv.

Men; vi må også legge til et par ting, vi må ha:

- Et “basistilfelle” - der rekursjonen skal stoppe.
- Noe som forandrer seg i kallene - slik at vi til slutt når basistilfellet.

Eksempel

```
class Rekursjonstest {
  public static void main(String[] args) {
```

```
          skrivTall(5);
  }

  public static void skrivTall(int n){
          if (n < 0) {
             return;
          }
          System.out.println(n);
          skrivTall(n-1);
  }
}
```

Oppgave 2

Bruk en lenkeliste fra tidligere til å løse de følgende oppgavene

(Semestersiden → Gruppe 5 → Uke 7 → FIFOListe.java)

2a)

Skriv ut elementene i lenkelisten først i riktig, så i motsatt rekkefølge ved hjelp av rekursjon. Hvor mye forskjell trenger det være i disse to metodene?

2b)

Finne størrelse på lenkeliste med rekursjon.

Testprogram (fra forrige uke + noe ekstra)

```
class TestFifoListe {
  //Teste Fifo-listen:
    public static void main(String[] args) {
      FIFOListe<String> liste = new FIFOListe<>();
      liste.leggTil("Hei");
      liste.leggTil("paa");
      liste.leggTil("deg");
      liste.leggTil("!");
      liste.skrivUtListe();

      System.out.println("Iterator-utskrift:");
      for(String t: liste){
           System.out.println(t);
```

```
      }

      liste.skriv_rek_sist();
      System.out.println();
      liste.skriv_rek_forst();

      System.out.println("ANT NODER: " + liste.antNoder());

    }
}
```