# Tekstanalyse

Obligatorisk oppgave 6 i INF1000

Frist 22.10.2014 kl 14.00

**Sammendrag**

Vi skal utvikle verktøy for enkel analyse av tekster som for eksempel å finne ut hvilke ord som forekommer ofte.

## Deloppgave 6.1

Vi skal jobbe med ord, så vi trenger et redskap for det: `class Ord` som representerer et ord i teksten; et ord kan forekomme én eller flere ganger.

Klassen skal ha følgende grensesnitt:

```
Konstruktør:
  Ord(String o) { ... }
    oppretter et Ord-objekt av den gitte teksten.

    Eksempel:
      new Ord("utmark")

Metoder:
  public String toString() { ... }
    returnerer ordet.

    Eksempel:
      new Ord("skog").toString()  gir  "skog"

  int hentAntall() { ... }
    henter data om hvor mange ganger ordet forekommer.

    Eksempel:
      Ord o = new Ord("grantre");
      o.hentAntall()  gir  1.

  void oekAntall() { ... }
    registrerer at ordet har forekommet en gang til.

    Eksempel:
      Ord o = new Ord("bjerk");
      o.oekAntall();
      o.hentAntall()  gir  2.
```

Skriv ferdig klassen Ord. Lag også et lite testprogram som viser at klassen fungerer. (Dette testprogrammet behøver ikke være stort; noe à la eksemplet over for `oekAntall` er tilstrekkelig.)

Synes du denne oppgaven er vanskelig? Se øvingsoppgaver 6.1.1, 6.1.2 og 6.1.3.

## Deloppgave 6.2

Vi trenger også en liste over alle ordene som forekommer i en bok: `class Ordliste`.
Klassen har følgende grensenitt:

```
Metoder:
  void lesBok(String filnavn) throws Exception { ... }
    leser alle ordene i en fil og legger dem inn i ordlisten.
    Filen må være organisert slik at det ligger akkurat ett ord på hver
    linje i filen.

    Eksempel:
      Ordliste ol = new Ordliste();
      ol.lesBok("scarlet.text");

  void leggTilOrd(String s) { ... }
    legger inn s som et nytt ord i ordlisten hvis det ikke finnes fra
    før. Hvis ordet allerede er der, skal antallet forekomster økes
    med 1.

    NB! Her regnes store og små bokstaver som like, så "asp", "Asp" og
    "ASP" er alle samme ordet.

    Eksempel:
      ol.leggTilOrd("London");

  Ord finnOrd(String s) { ... }
    finner et gitt ord s i ordlisten. Hvis ordet ikke finnes, får vi
    null som svar.

    Eksempel:
      if (ol.finnOrd("Edinburgh") == null) { ... }

  int antallOrd() { ... }
    forteller hvor mange ulike ord det finnes i ordlisten.

    Eksempel:
      if (ol.antallOrd() > 0) { ... }

  int antallForekomster(String s) { ... }
    finner ut hvor mange ganger ordet s forekommer i ordlisten.

    Eksempel:
      if (ol.antallForekomster("Watson") >= 100) { ... }

  Ord vanligste() { ... }
    finner det ordet som forekommer oftest i boken. Hvis det er flere
    ord som forekommer flest ganger, holder det å finne ett av dem.

    Eksempel:
      System.out.println("Vanligste ord er " + ol.vanligste());

  [Frivillig:]
  Ord[] alleVanligste() { ... }
    finner det eller de ordene som forekommer flest ganger i boken.
```

**Hint** Du kan lagre innholdet i ordlisten på den måten du foretrekker (for eksempel en array eller et `ArrayList`-objekt eller noe annet). Du kan anta at det aldri er mer enn 10 000 ulike ord i en bok.

Skriv ferdig klassen `Ordliste` og lag også et lite testprogram.

Synes du denne oppgaven er vanskelig? Se øvingsoppgaver 6.2.1, 6.2.2 og 6.2.3.

## Deloppgave 6.3

Vi skal teste opplegget vårt på den første Sherlock Holmes-boken *A study in scarlet* av Sir Arthur Conan Doyle. Last ned filen `scarlet.text` som inneholder boken formatert riktig (dvs ett ord på hver linje).

Lag et program `Oblig6` som benytter klassene `Ord` og `Ordliste` til å beregne og skrive ut følgende informasjon om denne boken:

a) Hvor mange ulike ord er brukt i boken?

b) Hvor mange ganger er ordet *Holmes* brukt?

c) Hvor mange ganger er ordet *elementary* brukt?

d) Hvilket ord er brukt flest ganger?

**Hint** Det tar noen sekunder å sjekke hele boken, så du kan spare mye tid på å lage en kort testfil du bruker til programmet er ferdig.

Synes du denne oppgaven er vanskelig? Se øvingsoppgaver 6.3.1, 6.3.2 og 6.3.3.

## Fremgangsmåte for innleveringer

1. Lag en fil som heter README.txt. Følgende spørsmål skal være besvart i filen:
   - Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
   - Hvor lang tid (ca) brukte du på innleveringen?
2. Logg inn på Devilry.
3. Lever alle Java-filene og README.txt.
4. Husk å trykke lever og sjekk deretter at innleveringen din er komplett.

## Ifis regler om obligatoriske oppgaver og andre innleveringer[1]

Ved alle pålagte innleveringer av oppgaver ved Ifi, enten det dreier seg om obligatoriske oppgaver, hjemmeeksamen eller annet, kreves det at arbeidet er et resultat av studentens egen innsats.

[1] Disse reglene er hentet fra http://www.mn.uio.no/ifi/studier/admin/obliger/.

### Krav til innleverte oppgaver

Å utgi andres arbeid for sitt eget er både ulovlig og uetisk og vil medføre sterke reaksjoner fra IFIs og Universitetets side, for eksempel utvisning i ett eller flere semestre; se *Rutiner for behandling av fuskesaker* på http://www.uio.no/om/regelverk/studier/studier-eksamener/fuskesaker/.

Vær derfor oppmerksom på:

- Hvis du tar med tekst, programkode, illustrasjoner og annet som andre har laget, må du tydelig merke det og angi hvor det kommer fra.
- Det er greit å få hint om hvordan en oppgave kan løses, men dette skal eventuelt brukes som grunnlag for egen løsning og ikke kopieres uendret inn.
- Du kan bli innkalt til samtale om dine innleveringer. Du må da kunne forklare innholdet i detalj og redegjøre for hvorledes det innleverte arbeidet er blitt til. Dersom samtalen avdekker at oppgaven ikke er et selvstendig arbeid og/eller du ikke kan gjøre rede for det innleverte arbeidet ditt, kan faglærer velge å ikke godkjenne oppgaven, også selv om gruppelærer har godkjent oppgaven i forkant av samtalen.

### Gruppearbeid

I noen emner skal det leveres gruppearbeid. Ifi krever da at alle medlemmer av gruppen kan gjøre rede for hovedtrekkene i det innleverte arbeidet. Dessuten må alle ha utført en rimelig del av det hele, og kunne identifisere og gjøre rede for i detalj sin del.

### Samarbeid ved individuelle innleveringer

Disse kravene betyr ikke at Ifi fraråder samarbeid —– tvert imot. Ifi oppfordrer studentene til å utveksle faglige erfaringer. I tilfeller hvor det skal leveres individuelle oppgaver er følgende viktig:

- Du kan diskutere en løsning sammen med andre, men dere skal ikke dele noen deler av løsningen (f eks ved å levere inn lik kode hvor kun variabelnavn er byttet ut).
- Dersom du tar med tekst, programkode, illustrasjoner og annet som andre har laget, må du tydelig merke det og angi hvor det kommer fra — i en selvstendig oppgave er dette noe som sjelden skal forekomme. Hvis du er i tvil om hva som er lovlig samarbeid, må du kontakte gruppelærer eller faglærer.

### Retningslinjer for obligatoriske oppgaver

En obligatorisk oppgave er en oppgave som må besvares og godkjennes for å kunne gå opp til avsluttende eksamen i et emne. Besvarelsen på en obligatorisk oppgave vurderes til godkjent eller ikke godkjent og kan ikke inngå i karaktervurderingen i emnet.

Antall obligatoriske oppgaver som gis i et emne samt tidspunktet for offentliggjøring og innlevering av oppgavene skal være klart på semestersiden for emnet ved studiestart.

Av oppgaveformuleringen skal det gå tydelig frem hva som forventes av innleveringen. Den faglige bakgrunnen som er nødvendig for å kunne løse oppgaven må være forelest eller gjort tilgjengelig på annen måte i god tid før innleveringsfristen.

*Retningslinjene i sin helhet* finnes på https://www.mn.uio.no/ifi/studier/admin/obliger/oblig-retningslinjer.html.