INF1000 - Obligatorisk innlevering 6

Frist: 16. Oktober kl 22:00

Tema denne uka: *Et første objektorientert program*.

Målet med oppgaven er å utvikle et verktøy for enkel analyse av tekster som vi leser fra fil. Vi ønsker for eksempel å finne ut hvilke ord som forekommer ofte i en gitt tekst. Vi vil benytte oss av objekter av en klasse `Ord`, som i tillegg til en tekststreng som representerer ordet selv, vil kunne holde på data om hvor mange forekomster det er av dette ordet i teksten. Vi skal også bruke en klasse `Ordliste`. Ordlisteklassen skal holde oversikt over alle ordene (objekter av klassen `Ord`) som forekommer i teksten.

Deloppgave 6.1

Vi skal jobbe med ord, så vi trenger et redskap for det: Objekter av `class Ord` skal representere hvert sitt unike ord fra teksten. Når vi leser inn teksten og kommer til et ord vi ikke har lest før lager vi et nytt objekt som husker ordet, og at det er funnet én gang. Når vi kommer til et ord vi har lest før, øker vi bare antall forekomster i det objektet vi allerede har laget for dette ordet.

Når teksten er ferdig lest inn i programmet vårt kan vi begynne å bruke de verktøyene vi har laget. Ved at like ord i teksten kun lagres én gang i datastrukturen vår sparer vi plass i datamaskinens minne under kjøring, og slipper å telle forekomster på nytt hver gang vi vil vite antall ganger et ord forekommer.

Klassen Ord skal ha følgende grensesnitt:

Konstruktør:

```
Ord(String tekst)
  oppretter et Ord-objekt av den gitte teksten.
```

Metoder:

```
public String toString()
  returnerer ordet.

public int hentAntall()
  henter data om hvor mange ganger ordet forekommer.
```

```
public void oekAntall()
  registrerer at ordet har forekommet en gang til.

public int hentLengde()
  returerer lengden til ordet

public int plassiDokument()
  basert på antall forekomster av ordet og dets størrelse,
  returnere hvor mye plass ordet tar i tekstdokumentet vi leser
```

For kjøretidseksempler til grensesnittet beskrevet over, se vedlegg A bakerst i oppgavesettet.

Skriv ferdig klassen Ord. Lag deretter et testprogram som viser at klassen fungerer som beskrevet i kjøreeksemplene. Testprogrammet ditt skal minimum opprette to objektreferanser og objekter av typen Ord, øke antallet på det ene objektet, og skrive ut ordet og antallet for begge objektene.

Synes du denne oppgaven er vanskelig? Se øvingsoppgaver 7.01 og 7.02.

Deloppgave 6.2

Vi trenger også en liste over alle ordene som forekommer i en bok: `class Ordliste`. Klassen skal følgende grensenitt: (Pass på at du har samme returverdier og public/private deklarasjon i ditt program, som det er definert i grensesnittet her.)

Metoder:

```
public 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.

private void leggTilOrd(String ord)
  legger inn 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.

public Ord finnOrd(String tekst)
  finner og returnerer et gitt ord s i ordlisten. Hvis ordet ikke finnes, får vi
  null som svar.

public int antallOrd()
  finner ut og returnerer hvor mange ulike ord det finnes i ordlisten.

public int antallForekomster(String tekst)
  finner ut og returnerer hvor mange ganger ordet "tekst" forekommer i ordlisten.
```

```
public Ord[] vanligste5()
  finner og returnerer de 5 ordene som forekommer oftest i boken.
  Hint: Det kan være lurt å ha en egen metode som sjekker om et Ord finnes i en Ord-array.

public Ord finnLengste()
  finner og returnerer det ordet ordlisten som er lengst

public Ord finnKorteste()
  finner og returnerer det ordet ordlisten som er kortest

public Ord tarMestPlassIDokument()
  finner og returnerer det ordet ordlisten som tar opp mest plass i dokumentet

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

Du *skal* benytte deg av en ArrayList<Ord> for å oppbevare ord i ordlisten. For kjøretidseksempler til grensesnittet beskrevet over, se vedlegg B bakerst i oppgavesettet.

Skriv nå ferdig klassen `Ordliste` og lag et program som tester at alle funksjonalitetene beskrevet over fungerer slik de skal.

Synes du denne oppgaven er vanskelig? Se øvingsoppgaver 7.03 og 7.04.

Deloppgave 6.3

Vi skal teste programmet vårt på den første Sherlock Holmes-boken *A study in scarlet* av Sir Arthur Conan Doyle. Last ned filen `scarlet.txt` 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 forekommer i boken?

b) Hvor mange ganger forekommer ordet *Watson*?

c) Hvor mange ganger forekommer ordet *elementary*?

d) Hvilke 5 ord forekommer flest ganger?

e) Hvilket ord er lengst?

f) Hvilket ord er kortest?

g) Hvilket ord tar opp mest plass i dokumentet?

Tips: 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.

Vedlegg A - kjøreeksempler til klassen Ord

Konstruktør:

```
new Ord("utmark")
```

tostring():

```
new Ord("skog").toString() returnerer  "skog"
```

hentAntall():

```
Ord forsteOrd = new Ord("grantre");
forsteOrd.hentAntall()  gir  1.
```

oekAntall():

```
Ord andreOrd = new Ord("bjerk");
andreOrd.oekAntall();
andreOrd.hentAntall()  gir  2.
```

hentLengde():

```
Ord tredjeOrd = new Ord("furu");
tredjeOrd.hentLengde() gir 4
```

plassIDokument():

```
Ord fjerdeOrd = new Ord("eik");
fjerdeOrd.oekAntall();
fjerdeOrd.oekAntall();
tredjeOrd.plassiDokument() gir 6
```

Vedlegg B - kjøreeksempler til klassen Ordliste

lesBok():

```
Ordliste liste = new Ordliste();
liste.lesBok("scarlet.text");
```

leggTilOrd():

```
liste.leggTilOrd("London");
```

finnOrd():

```
if (liste.finnOrd("Edinburgh") == null) { ... }
```

antallOrd():

```
if (liste.antallOrd() > 0) { ... }
```

antallForekomster():

```
if (liste.antallForekomster("Watson") >= 100) { ... }
```

vanligste5():

```
System.out.println("De 5 vanligste ordene er:");
int[] vanligste = liste.vanligste5();
for (int i = 0; i < 5; i++){
    System.out.println("- " + vanligste[i]);
}
```

finnLengste():

```
Ordliste ordliste = new Ordliste();
ordliste.leggTilOrd("furu");
ordliste.leggTilOrd("eik");
ordliste.finnLengste() gir Ord-objektet til "furu"
```

finnKorteste():

```
Ordliste ordliste = new Ordliste();
ordliste.leggTilOrd("furu");
ordliste.leggTilOrd("eik");
ordliste.finnLengste() gir Ord-objektet til "eik"
```

tarMestPlassIDokument():

```
Ord mestPlass = ordliste.tarMestPlassIDokument();
System.out.println("Tar mest plass i dokumentet:" + mestPlass.toString());
```

Krav til innleveringen

1. Klassenavnet og filnavnet skal være identisk.
2. Klassenavn skal skrives med stor forbokstav.
3. Variabelnavn skal ha liten forbokstav.
4. Oppgaven må kunne kompilere og kjøre på IFI sine maskiner.
5. Kun .java-filen skal innleveres.
6. Ikke bruk æ, ø eller å i .java-filene(heller ikke som kommentarer eller utskrift).
7. Filene skal inneholde gode kommentarer som forklarer hva programmet gjør.
8. Programmet skal inneholde gode utskriftssetninger som gjør det enkelt for bruker å forstå.
9. Metodenavn skal skrives med liten forbokstav.
10. Koden skal være riktig indendert. Er du usikker, se Appendix J i Big Java.
11. Hver klasse skal ligge i sin egen .java-fil.

Fremgangsmåte for innleveringer i INF1000

1. Lag en fil som heter README.txt. Følgende spørsmål skal være besvart i filen:
   (a) Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
   (b) Hvor lang tid (ca) brukte du på innleveringen?
   (c) Var det noen oppgaver du ikke fikk til? Hvis ja:
   - Hvilke(n) oppgave er det som ikke fungerer i innleveringen?
   - Hvorfor tror du at oppgaven ikke fungerer?
   - Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?
2. Logg inn på Devilry.
3. Lever de 3 .java-filene (Ord.java, Ordliste.java, Oblig6.java), og eventuelle andre testprogrammer du har laget i *samme innlevering*.
4. Husk å trykke lever og sjekk deretter at innleveringen din er komplett.

Den obligatoriske innleveringen er minimum av hva du bør ha programmert i løpet av en uke. Du finner flere oppgaver for denne uken her.