Datastrukturer og algoritmer
Henning Christiansen

Opgave til løsning ved øvelserne 10. september 2002.

Opgavens eksempel er en simpel form for datamining på tekstfiler baseret på optælling af sekvenser af ord. Formålet med opgaven er at diskutere og arbejde med, hvordan de begreber det handler om kan omsættes til et fornuftigt klassihierarki og et fornuftigt Java-program.

Spørgsmål 1

Der skal designes og implementeres en klasse til at repræsentere ord; kald den class word. Det gælder her, at vi vil undgå at repræsentere hver forekomst af en tekststreng mange gange. I stedet repræsenteres hvert ord ved et *heltal*: Derved bliver sammenligning af ord langt mere effektivt, dvs. at en "equals" metode kan implementeres ved at sammeligne to tal i stedet for at tælle sig ned gennem en streng.

Anbefalet strategi: Klassen word har en hukommelse af de ord, vi har set indtil nu. Hvert ord tildeles en unikt nummer *n*. Ud fra *n* kan vi finde tilhørende tekststreng

Eksempel:

```
new word("sildepostej")
```

Der oprettes et objekt; hvis "sildepostej" er registreret allerede, så med den *n*-værdi som hører til "sildepostej"; ellers registreres "sildepostej" det næste ubrugte nummer.

NB: Vi ved godt, at det ikke er særligt effektivt at søge sekventielt gennem en usorteret rækkefølge, men fidusen ved et godt struktureret program er, at vi altid kan udskifte den del af programmet, når vi en gang har lært om strategier til effektiv lagring og genfinding.

Implementér klassen word med konstruktorer, equals, og toString-metoder. Benyt ArrayList til hukommelsen.

Spørgsmål 2

Klassen word skal bruges i et program, der optæller antallet af gange ord eller sekvenser af ord forekommer i en tekst. Vi sætter en grænse på sekvenser op til 4 ord, altså, vi er interesserede i at tælle antal gange en sekvenser af længde 1 (svarende til ord), og af længde 2, 3 og 4. For eksemplet "a b c d b c d a a" svarer det til følgende:

a: 3	c: 2	d b c d: 1
a b: 1	c d: 2	b c d a: 1
a b c: 1	c d b: 1	c d a: 1
a b c d: 1	c d b c: 1	c d a a: 1
b: 2	d: 2	d a: 1
b c: 2	d b: 1	d a a: 1
b c d: 2	d b c: 1	a a: 1
b c d b: 2		

Dette giver tydeligvis alt for meget information, så vi sætter en nedre grænse på, hvor mange gange ord en sekvens skal forekomme før det er interessant; sætter vi den til 3
bliver der i eksemplet ovenfor kun sekvensen "a" tilbage (men vi er jo nødt til at tælle dem alle med fra starten og derefter tynde ud).

Skriv klasser til at repræsentere sekvenser og tælleværk. Benyt igen ArrayList (vel vidende at det ikke er det mest effektive i verden) til hukommelsen.
Diskutér hvad der er mest hensigtsmæssigt:
én klasse for samtlige sekvenser eller en overklasse med fire underklasser.

Vedlagt link til en fil med "Den grimme ælling" i engelsk oversættelse, kun med små bogstaver og alle punktuationstegn fjernet, og filen slutter med ordet "endoffile".
Brug ovenstående til at skrive et program som optæller og udskriver alle sekvenser som forekommer mere end fire gang.

Spørgsmål 3

Indenfor den type tekstprocessering udskiller man ofte en gruppe ord kaldet stopord (en lidt misvisende betegnelse). Stopord er ord, som i sig selv ikke bærer meget betydning, men blot klistrer andre ord sammen. F.eks. "in", "on", "an".

Vedlagt link til en fil med stopord, som slutter med ordet endoffile.

Diskutere, hvordan man kan indarbejde begrebet stopord i vores klassehierarki; det er meningen, at de faktiske stopord skal læses ind fra omtalte fil, når et program, som bruger dem startes.

Hvis tid: tilpas programmet således at det frasorterer (dvs. ikke medregner) sekvenser som starter med eller slutter med et stopord, men ellers fungerer uændret.