Tekstanalyse, obligatorisk oppgave 3 i INF1000 - høsten 2013

Innleveringsfrist: fredag 11. oktober kl. 23.59 i Devilry

I denne oppgaven skal du analysere en tekst som foreligger på en fil. Til denne oppgaven har vi laget en versjon av 'Alice's adventures in Wonderland' av Lewis Carroll på filen:'Alice.txt' som du skal nytte i testkjøringene, men programmene dine skal selvsagt virke på alle tekstfiler (Se vedlegg I om flere opplysninger om filen 'Alice.txt'). I vedlegg II er en oppskrift på hvordan du skriver begynnelsen av programmene dine og får tak i et filnavn når du starter.

De programmene det spørres om her skal du skrive i Java (Oblig3A.java, Oblig3B.java, Oblig3C.java og eventuelt også Oblig3D.java) Hver deloppgave skal besvares for seg. Når en deloppgave bygger videre på en tidligere deloppgave, skal du levere komplette kompilerbare og kjørbare programmer for hver deloppgave (hvilket f.eks. betyr at svaret du leverer på deloppgave B nedenfor vil være en utvidelse/modifikasjon av svaret du leverer på deloppgave A).

A) Skriv et program som leser en tekst fra en tekstfil og som:

- Finner hvilke ord som forekommer i teksten.
  Du skal lese filen ord for ord (f.eks med inWord() eller next() i easyIO).
- Teller opp hvor mange ganger hvert ord forekommer. NB: du kan anta at det maksimalt er 5000 unike (forskjellige, ulike) ord på filen som leses.
- Skriver en oppsummering ut til en ny fil "oppsummering.txt". Oppsummeringen skal starte med at du først skriver ut hvor mange ord det totalt ble lest og antall unike ord du finner, for eksempel slik:

  ```
  Antall ord lest: 30900 og antall unike ord: 3777.
  ```

  (hvis det er antallene du finner)
- Deretter kommer en rad for hvert unike ord i teksten som ble lest inn, med ordet og antall ganger det forekommer. Rekkefølgen ordene skrives ut i er vilkårlig. Filen kunne f.eks. starte slik (dette er bare et eksempel, din oppsummering kan se annerledes ut):

  ```
  cake   4
  a      320
  piece  2
  of     24
  ```

  ....osv....
- Filen som leses skal være parameter til programmet når det startes opp (>java Oblig3A <teksfil>). På oblig-området ligger to tekstfiler – en med Alice og en med tre norske folkeeventyr. Du skal teste på begge disse to, men det du innleverer (oppsummering.txt) skal være fra Alice.txt. Se vedlegg II.
- To ord betraktes som like selv om en av dem har stor forbokstav eller bare store bokstaver, og den andre ikke. Du må derfor ikke sammenligne to tekster s og t direkte, men nytte den innebygde metoden toLowerCase() i klassen String og sammenligne t. toLowerCase () og s. toLowerCase () (du vet ikke hvilken av dem som har stor førstebokstav eller bare store bokstaver).

Oppgaven kan løses på mange forskjellige måter (f.eks. kunne vi brukt HashMap, men det er ikke forelest enda, og skal ikke nyttes). Her skal du benytte en String-array **`ord`** til å holde rede på hvilke ord som er lest fra fil og en tilhørende (like lang) int-array **`antall`** til å telle opp hvor mange ganger hvert ord forekommer (vi skal for eksempel ikke bruke HashMap som enda ikke er forelest). Du må også ha en teller **`antUnikeOrd`** som holder rede på hele tiden hvor mange unike ord du hittil har lest inn.

Når du leser inn et ord fra fil og samme ord ikke har vært lest tidligere, skal du legge inn ordet på første ledige plass i arrayen **`ord`**, og øke tilhørende verdi i arrayen **`antall`** med 1, og tilsvarende må telleren **`antUnikeOrd`** da også økes med 1. Når du leser inn et ord fra fil og samme ord har vært lest inn tidligere (du leser f.eks. ordet 'cake' for annen gang) så skal du ikke legge inn noe nytt ord i String-arrayen, men du må huske å oppdatere arrayen **`antall`**.

Organiser programmet slik at du har en klasse **`class Oblig3A`** som inneholder en main-metode og som lager et objekt av en annen klasse **`class OrdAnalyse`** og deretter kaller på en passende (objekt-) metode i denne klassen som leser inn tekstfilen, teller opp og lager oppsummeringsfilen. La de to arrayene være objektvariable i klassen **`OrdAnalyse`** og innfør nye objektvariable og metoder etter behov.

B) I denne deloppgaven skal du ta utgangspunkt i programmet du skrev i deloppgave A og utvide med en ny funksjonalitet: å finne de ordene som forekommer flest ganger, og helt konkret er oppgaven å først finne hvor mange ganger det mest brukte ordet forekommer, og så skrive ut ***alle*** ord som forekommer minst 10% av det antall ganger som det ordet som forekommer flest ganger (du vil bli forbauset over hvor få ord dette er i forhold til alle ordene du har lest). Dette kan passende gjøres etter at du har lest hele tekstfilen og talt opp hvor mange ganger hvert ord forekommer. Siden du ikke har sortert ordene dine vil de komme ut i 'tilfeldig' rekkefølge. Svaret skal du skrive ut på skjermen, f.eks. :

```
Vanlige ord: she    (539 forekomster)
Vanlige ord: Alice  (369 forekomster)
Vanlige ord: the   (1628 forekomster)
……..…..
```

For å få til dette kan det lønne seg å først finne den største verdien i arrayen **`antall`** og så gå gjennom arrayen en gang til og plukke ut alle ordene som forekommer minst 10% så mange ganger.

C) (Vanskelig) Du skal nå utvide programmet til også å se på ord-par, altså ord som forekommer rett etter hverandre. I denne teksten (når vi betrakter punktum og komma som egne ord):

A rabbit looked up from its hole. Where am I, it wondered.

forekommer følgende fjorten ord-par(her med små bokstaver skilt med -):

a - rabbit	rabbit - looked	looked - up	up - from	from - its
its - hole	hole - .	. - where	where - am	am - i
i - ,	, - it	it - wondered	wondered - .	

Når du leser tekstfilen skal du nå (i tillegg til alt det som gjøres under punkt A og B) også telle opp antall forekomster av hvert enkelt ordpar. Det er flere måter dette kunne vært gjort på , men i denne oppgaven er vi ute etter en bestemt måte for å trene på bruk av 2D-arrayer. Du skal benytte en array **`int[][]antallPar`** som har like mange rader og like mange kolonner som antall unike ord i teksten som er lest inn (hvis teksten var "det er det jeg sier" ville altså 2D-arrayen ha fire rader og fire kolonner). I utgangspunktet er alle verdier i arrayen satt til 0. Etterhvert som tekstfilen leses, skal arrayen oppdateres slik at når hele filen er lest så er **`antallPar[i][j]`** antall forekomster av ordparet **`ord[i]-ord[j]`** for alle i og j.

For å illustrere dette, tenk deg at teksten på filen er "you are you are". Da blir

ord[0]: you antall[0]: 2
ord[1]: are antall[1]: 2

og

antallPar[0][0]: 0 antallPar[0][1]: 2
antallPar[1][0]: 1 antallPar[1][1]: 0

Som en test skal du skrive ut på skjermen alle de ulike ordene som har etterfulgt ordet 'Alice' i den innleste teksten. *Hint*: Anta at Alice er ord 'i'. Da er alle etterfølgere til 'Alice' alle de: **`ord[j]`**, hvor **`antallPar[i][j]`** > 0.

D) (Frivillig ekstraoppgave). Her skal du bruke dataene du fant om antallPar i del C) til å skrive en tilfeldig novelle med 500 ord med disse ordene som følger:

1. Velg et tilfeldig ord t i den innleste teksten (husk å si :import java.util.*;) slik:

   Random r = newRandom();
   int i = r. nextInt(antallOrd); // velger et tilfeldig tall mellom 0 og antallOrd-1
   String t = ord[i]; Skriv ut ordet t.

2. Velg et tilfeldig antallPar hvor t er det første ordet. Dette gjør du ved å velge blant de antallParene i arrayen **`antallPar[i][]`** (hvor i er definert som i pkt 1)
   Hint: Du vet at alle ord i innlest tekst har like mange etterfølgere som det selv forekommer – unntatt det siste ordet som har én færre. Det tallet finner du i **`antall[i]`**, dvs at ordet t har forekommet i **`antallPar[i][]`** som første ord **`antall[i]`** ganger). Velg et tilfeldig tall x mellom 1 og **`antall[i`**]. Gå så langs **`antallPar[i][j]`** ved å telle ned x med det du finner i **`antallPar[i][j]`** for j=0,1,…inntil x <0. Den j'en hvor x ble ≤ 0 er indeksen i **`ord[]`** hvor du finner neste ord.- Kall det andre ordet s1, skriv ut s1.

3. Gjenta punkt 2) 499 ganger med hver gang ny s1 som t, og j fra pkt. 2 som ny i.

Du får da en tekst som i forbausende grad minner om noe fornuftig og den filen du har lest, men som egentlig bare er 'tull' Det er en novelle med samme ordforråd som originalen.

Vedlegg I – litt om filen 'Alice.txt'

I tillegg til vanlige bokstaver, inneholder en fil som Alice.txt en rekke skilletegn som punktum(.), komma(,) , spørsmålstegn og en del andre slike tegn. De skaper problemer når vi leser teksten – for eksempel si at vi har lest ordet **cat** inne i en setning, men så forekommer det samme ordet på

slutten av en setning, men da som: **cat.** - da vil enkel innlesning av de to ordene finne to ulike ord, noe vi ikke ønsker. Dette problemet har vi forsøkt løst ved å legge inn blanke før og etter alle slike skilletegn samt fjerne alle anførselstegn (‘ og «). Det medfører at alle skilletegnene blir egne ord og at komma, ‘the’ og punktum blir faktisk de tre hyppigste ordene på filen. Hvis du løser oppgave D ser du at det er bra at du av og til velger å sette komma og punktum i en tekst.

N.B. Analyserer du en fil med norsk tekst (som Folkeeventyr.txt) på en Windows-maskin, vil æ, ø og å bli galt/rart skrevet ut fordi kommandovinduet i Windows bruker et annet tegnsett enn det vi nå bruker på anlegget på Ifi (UTF-8). Faktisk vil du få to rare tegn ut for hver av de norske tegnene. Det er ikke din feil og ikke enkelt å gjøre noe med, så overse dette.

Vedlegg II – start av alle programmer:

Alle programmene dine (Oblig3A, Oblig3B, …), her et eksempel med deloppgave C, Oblig3C, har en enkel klasse med ‘main’ som bare sjekker at antall parametre er OK og lager et objekt av en annen klasse hvor den kaller en metode i det objektet. Dette viser også hvordan parameteren til ‘main’ brukes. Det er en String-array args, som pakker inn alt som er på linjen etter at du har kalt på java i kommando-vinduet. Skriver du der :

```
>java Oblig3C Alice.txt
```

så vil args-arrayen inneholde ett element som er denne Stringen: **`Alice.txt`**. Hadde du imidlertid skrevet: **`>java Oblig3C Alice.txt 12`**, så ville du fått to elementer i args, ‘Alice.txt’ og ‘12’ som hhv., args[0] og args[1] , osv . Merk at en ‘blank’ adskiller de ulike elementene på kall-linja.

```
class Oblig3C {
      public static void main(String[] args) {
            if (args.length < 1) {
                  System.out.println("Bruk: > java Oblig3C <tekstfil>");
            } else {
                  OrdAnalyseC oa = new OrdAnalyseC(args[0]);
                  oa.analyser();
            }
      }
}

class OrdAnalyseC {
… Her skriver du objekt-dataene og objekt-metodene dine  ..

}
```

--- slutt på Oblig3, lykke til ---