OBLIGATORISK OPPGAVE 2 - INF110, høsten 2003

1 Dikteroppgaven

I denne oppgaven skal dere lage en tekst (**utteksten**). Utteksten er en tilfeldig etteraping av hvordan det norske språket er anvendt i en annen seriøs tekst (**innteksten**). Etterapingen skjer ved:

> når et ord *Oi* er skrevet i utteksten, skal neste ord *Oi+1* velges ved trekning blant de ordene i innteksten som har etterfulgt ordet *Oi*.

Mer detaljert om dette kravet senere.

Hensikten med denne obligatoriske oppgaven er å trene bruk av grensesnitt (interface) og bruk av lister. Dette skal programmeres, og bruk av klassene i Java-biblioteket for lister vil ikke bli godkjent.

Programmet består av to faser:

- Innlesning av innteksten ord for ord med oppbygging av datastrukturene (listene). Vi har først en liste (i del 3 flere lister) over alle ordene. For hvert ord lages en egen liste over alle etterfølgerordene til dette ordet.
- Første ord, *O0*, i utteksten velges tilfeldig. For ord *Oi* skal neste ord *Oi+1* i utteksten velge ved trekning blant de ordene i innteksten som har etterfulgt ordet *Oi*. Valget skal skje med samme vekt på etterfølgerordene som antall ganger det har etterfulgt *Oi* i innfilen.

Kravet om valg av neste ord kan vises med et eksempel: Anta "jeg" er valgt som første ord. Hvis "er" har etterfulgt "jeg" 2 ganger og "spiser" 1 gang i innteksten, så skal neste ord velges ved tilfeldig trekning (class Random) med 67% (~2/3) sannsynlighet for "er" og 33% (~1/3) sannsynlighet for "spiser". Trekningen viser så hvilket av de to ordene som velges (si: "spiser"). Så betraktes ordene som følger "spiser" i innteksten, og neste ord trekkes tilfeldig blant disse med vekt osv...

Programmet i del 2 og 3 nedenfor skal startes med følgende parametere:

```
>java DiktProg 'innfil' 'utfil' 'maksordperlinje'
```

der "maksordperlinje" er maksimalt antall ord på hver linje i utteksten. Antall ord som skrives ut på hver linje skal velges tilfeldig, men ingen av linjene får lov til å overskride dette antallet.

Merk at del 2 og del 3 nedenfor representerer to varianter av oppgaven:

- del 2 er en enkel implementasjon av grensesnittet Dikter,
- del 3 er en noe mer effektiv implementasjon av det samme grensesnittet.

Del 1 - lag en dokumentert ADT

Det skal lages et grensesnitt (interface) **Dikter** med tre metoder:

- en metode for å behandle neste ord fra innteksten (inn i ordliste og inn i etterfølgerliste til forrige innleste ord),
- en metode for å velge første ord til utteksten,
- og en metode for å finne nesteord til utfila.

Grensesnittet programmeres som en "public interface" i en fil "Dikter.java". Det skal være *"javadoc"*- kommentarer til både grensesnittet og hver av de tre metodene. Dere skal så kjøre *"javadoc"* på "Dikter.java" og levere med html-siden utskrevet fra dette. Deloppgaven godkjennes dersom kommentarene kommer med på den genererte 'Dikter.html'- fila.

Del 2 - den enkle løsningen

I denne løsningen brukes *en* felles liste over ordene i innfilen (hvert ord bare en gang). Merk at Java betrakter store og små bokstaver som ulike. Videre vil punktum og komma kunne bli inkludert i ordene (avhengig av hvordan innlesningen skjer). Dere stilles fritt til hvor avansert dette behandles. En god løsning er:

- bokstavene omgjøres til små før ordet fra innfilen behandles,
- punktum og komma betraktes som egne ord,
- stor bokstav i ord etter punktum fikses i forbindelse med utskrift.

Det skal lages en klasse "LagDikt1" som implementerer grensesnittet **Dikter**. I tillegg skal det lages et komplett program for oppstart (behandle parameterne fra kommandolinja), innlesning og behandling av ordene fra "innfil", og generering av ord og utskrift til "utfil".

Tips:

- Klassen Random brukes til å trekke tilfeldige tall og metoden nextInt(n) gir et tilfeldig tall mellom [0 .. n-1] - grensene inklusive.
- Klassen Inn i pakken inf101, inneholder to metoder lastItem() og inString() som er bra egnet til å lese "innfil" ord for ord.
- Tilsvarende finnes det velegnede metoder i klassen Ut for å skrive "utfil".
- Husk hvordan man i Java sammenligner om to Stringer har samme innhold.
- For at også det siste ordet får en etterfølger, kan dere anta at det første ordet i "innfil" etterfølger det siste ordet.

Programmet kan testes på en liten testfil dere selv lager. Innleveringen består i at dere bruker fila: 'vildand.txt' som innfil. Denne laster dere ned i raden for uke 5 i "detaljert undervisningsplan", som er tilgjengelig fra fagets høst 2003 hjemmeside. Den inneholder Henrik Ibsens skuespill Vildanden. Fra denne genererer du en 500 ords utfil med maksimum 10 ord per linje, som skrives ut og innleveres sammen med Java-koden til programmet.

Del 3 - en forbedret og ”rask” løsning

Tekstfilen ”vildand.txt” er nær 32 000 ord lang (hvorav ca. 4000 ulike), og løsningen tar merkbar tid. Det er oppbyggingen av den lange listen av ordene som tar tid. Dere skal nå lage en ny implementasjon av grensesnittet ”Dikter” i en klasse ”LagDikt2”. Den lange ordlista erstattes med 256 lister. Et ord knyttes til en av listene avhengig av verdien av de 8 nederste bit av første bokstav i det gitte ordet.

Tips: En måte å få tak i denne verdien (anta String ord) er

```
int ind = ((int)ord.charAt(0)) % 256;
```

Lag Java-kode for ”LagDikt2” og kjør programmet på ’vildand.txt’. Mål hastighetsforbedringen fra LagDikt1. Lever koden sammen med en kortfattet forklaring på hvorfor programmet nå går merkbart raskere, men på ingen måte 256 ganger raskere.