Avdeling for allmenne fag

EKSAMEN

I

801 OBJEKTBASERT PROGRAMMERING

08.01.2003

Tid:	5 timar
Målform:	Bokmål / Nynorsk
Sidetal:	6 + framside
Hjelpemiddel:	Alt skriftleg materiale
Merknader:	Ingen
Vedlegg:	Ingen

Eksamensresultata blir offentleggjort på følgjande internettadresse:
http://www-bo.hit.no/af/ Klikk Student og så eksamen

eller du kan gå inn på Arena høgskole – da trenger du heller ikkje kandidatnummeret

Bokmål

Prosentsatsen ut for hver oppgave antyder vekt ved sensur. Les oppgaveteksten nøye, og forsøk å svare på så mange spørsmål som mulig.

Lykke til!

Oppgave 1 (40%)

I denne oppgaven skal du lage hjelpemetoder til bruk i et program for stavekontroll av dokumenter opp mot en ordliste. For ord som ikke finnes i ordlista skal programmet foreslå alternative ord som ligner. I den forbindelse er det hensiktsmessig å definere avstanden mellom to ord.

- For to ord som er like lange sier vi at avstanden mellom dem er antall posisjoner der de er forskjellige. For eksempel så er avstanden mellom "kake" og "koke" lik 1 fordi de er forskjellige i posisjon 1 (posisjoner i tekststrenger starter på 0). For to like ord er avstanden lik 0.
- For to ord som ikke er like lange sier vi at de er forskjellige i alle posisjoner utover slutten av den korteste tekststrengen. For eksempel så er avstanden mellom "kake" og "kokekunst" lik 6: avstanden er lik 1 fram til posisjon 4 (altså 0, 1, 2 og 3), men i tillegg inneholder ordet "kokekunst" 5 tegn i "forlengelsen" av ordet "kake".
- Det er videre nyttig å definere avstanden mellom to ord fram til en bestemt posisjon *pos*. Her ser vi kun på forskjeller mellom ordene fra og med posisjon 0 til posisjon *pos*. For eksempel så er avstanden mellom "kake" og "kokekunst" fram til posisjon 2 lik 1. Dette svarer til å finne avstanden mellom "kak" og "kok".

Lag en metode for å beregne avstanden mellom to tekststrenger *s* og *t* fram til posisjon *pos*.

Du kan anta at parameteren *pos* alltid er mindre enn eller lik lengden til den korteste av *s* og *t*.

```
public int avstandDelStreng(String s, String t, int pos)
```

Tips: Klassen *String* inneholder metoden *charAt* for å hente ut et tegn i en bestemt posisjon. Hvis *s* inneholder tekststrengen "kake", så vil *s.charAt(1)* returnere *char*-verdien 'a'.

1-b)

Lag en metode for å beregne avstanden mellom to tekststrenger *s* og *t*. Du skal nå ikke anta at *s* og *t* er like lange.

```
public int avstand(String s, String t)
```

Tips: Bruk metoden du laget til deloppgave 1-a. Dette kan du gjøre selv om du ikke klarte å løse denne deloppgaven.

1-c)

Lag en metode som avgjør om en tekststreng *s* finnes i en tekststreng-tabell *ordliste*.

```
public boolean finnes(String s, String[] ordliste)
```

Lag en metode som gitt en tekststreng *s* og en tekststreng-tabell *ordliste* finner en tekststreng i tabellen med kortest mulige avstand til *s*. Det kan finnes flere tekststrenger med like kort avstand. Metoden kan da fritt velge en av disse, for eksempel den første.

```
public String lignerMest(String s, String[] ordliste)
```

Oppgave 2 (20%)

2-a)

Det er ønskelig å legge modifiserte versjoner av metodene du laget til oppgave 1 inn i en modellklasse *Ordliste*. Denne klassen bør ha selve ordlista (en tekststreng-tabell) som en privat instansvariabel.

Når man oppretter objekter av klassen *Ordliste* bør man kunne overføre navn på tekstfilen der ordlista er lagret som parameter til konstruktøren. Du kan anta at tekstfilen inneholder ett ord på hver linje, og at antall ord er lagret på første linje.

Lag klassen *Ordliste* med konstruktør og instansvariable. Du trenger ikke gjenta koden til metodene fra oppgave 1, men få fram eventuelle endringer i parametre og returverdier.

2-b)

Du skal forklare hvordan man kan programmere en modellklasse *Orddeler* for å hente ut ett og ett ord fra en tekstfil. Du skal altså ikke skrive kode, men legg vekt på å få fram hvilke instansvariable du vil trenge, og forklar hvordan hver metode kan programmeres. Klassen skal være bygget opp slik:

```
public class Orddeler extends Object {

  // Deklarasjon av nødvendige instansvariable

  public Orddeler(String filnavn) {  }
  public boolean flereOrd() {  }
  public String nesteOrd() {  }
}
```

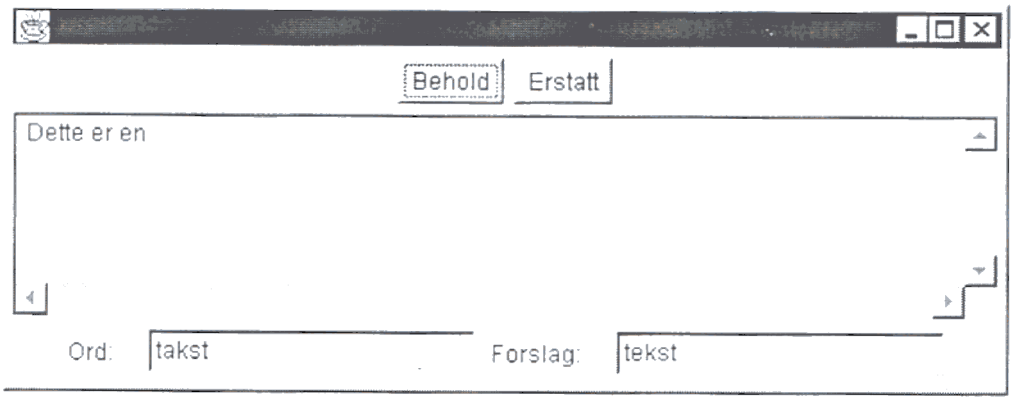
Forklaring til metodene:

- Konstruktøren: Når man lager et objekt av klassen *Orddeler* blir filnavnet overført som parameter. For å forenkle programmeringen antar vi at punktum og blank er eneste skilletegn mellom ord.

- Metoden flereOrd: Det skal være mulig å hente ut ett og ett ord fra en orddeler. Denne metoden skal returnere *true* hvis det er flere ord å hente ut, og ellers *false*.
  Metoden nesteOrd: Et kall på denne metoden skal returnere neste ord fra tekstfilen. Hvis tekstfilen inneholder setningen "Dette er en tekst." skal fire påfølgende kall på metoden returnere "Dette", "er", "en" og "tekst".

Oppgave 3 (40%)

Du skal nå lage en applikasjon for stavekontroll ved å bruke modellklassene *Ordliste* og *Orddeler*. Brukergrensesnittet skal se slik ut:



Du kan anta at ordlista er lagret med navn "ordliste.txt" og at filen som skal stavesjekkes er lagret med navn "innfil.txt", begge i samme katalog som det kjørbare programmet er lagret. Tekstboksen nede til venstre inneholder et ord fra innfilen, og tekstboksen nede til høyre inneholder til enhver tid det ordet fra ordlista som ligner mest.

Programmet skal ha følgende funksjonalitet:

- Ved oppstart skal det opprettes et *Ordliste*-objekt og et *Orddeler*-objekt. I den forbindelse blir tekstfilene lest inn.
- Ved oppstart skal programmet dessuten hente det første ordet fra innfilen og det ordet fra ordlista som ligner mest, og vise disse i tekstboksene nederst i vinduet.
- Ved klikk i kommandoknappen merket *Behold* skal ordet fra innfilen legges til i det store tekstområdet midt i vinduet. Ved klikk i kommandoknappen *Erstatt* skal ordet fra ordlista legges til i stedet. I begge tilfeller skal programmet deretter hente neste ord fra innfilen og det ordet som ligner mest fra ordlista, og sette disse inn i tekstboksene for ny behandling av bruker.
- Når det ikke er flere ord å hente fra innfilen skal de to tekstboksene renskes. Man kunne da tenke seg at den modifiserte teksten ble lagret til fil, men det skal du ikke programmere her.
- Ord som er i ordlista skal programmet automatisk legge til i tekstområdet uten at bruker må klikke *Behold*.

Lag klassen *Stavesjekk* som definerer brukergrensesnittet og kaller på de riktige metodene modellklassene *Orddeler* og *Ordliste*.

Nynorsk

Prosentsatsen ut for kvar oppgåve syner vekt ved sensur. Les oppgåveteksten nøye, og freist å svare på så mange spørsmål som mogleg.

Lykke til!

Oppgåve 1 (40%)

I denne oppgåva skal du lage hjelpemetodar til eit program for stavekontroll av dokument opp mot ei ordliste. For ord som ikkje fins i ordlista skal programmet foreslå alternative ord som liknar. Det er då nyttig å definere avstanden mellom to ord.

- For to ord som er like lange seier vi at avstanden mellom dei er talet posisjonar der dei er forskjellige. Til dømes så er avstanden mellom "kake" og "koke" lik 1 fordi dei er forskjellige i posisjon 1 (posisjonar i tekststrengar startar på 0). For to like ord er avstanden lik 0.
- For to ord som ikkje er like lange seier vi at dei er forskjellige i alle posisjonar utover slutten av den kortaste tekststrengen. Til dømes så er avstanden mellom "kake" og "kokekunst" lik 6: avstanden er lik 1 fram til posisjon 4 (altså 0, 1, 2 og 3), men i tillegg inneheld ordet "kokekunst" 5 teikn i "forlenginga" av ordet "kake".
- Det er vidare nyttig å definere avstanden mellom to ord fram til ein bestemt posisjon *pos*. Her ser vi berre på forskjellar mellom orda frå og med posisjon 0 til posisjon *pos*. Til dømes så er avstanden mellom "kake" og "kokekunst" fram til posisjon 2 lik 1. Dette svarar til å finne avstanden mellom "kak" og "kok".

1-a)

Lag ein metode for å berekne avstanden mellom to tekststrengar *s* og *t* fram til posisjon *pos*.

Du kan rekne med at parameteren *pos* alltid er mindre enn eller lik lengda til den kortaste av *s* og *t*.

```
public int avstandDelStreng(String s, String t, int pos)
```

Tips: Klassa *String* inneheld metoden *charAt* for å hente ut eit teikn i ein bestemt posisjon Viss *s* inneheld tekststrengen "kake", så vil *s.charAt(1)* returnere *char*-verdien 'a'.

1-b)

Lag ein metode for å berekne avstanden mellom to tekststrenger *s* og *t*. Du skal nå ikkje rekne med at *s* og *t* er like lange.

```
public int avstand(String s, String t)
```

Tips: Nytt metoden du laga til deloppgåve a. Dette kan du gjere sjølv om du ikkje klarte å løyse denne deloppgåva.

1-c)

Lag ein metode som avgjer om ein tekststreng *s* fins i ein tekststreng-tabell *ordliste*.

```
public boolean finnes(String s, String[] ordliste)
```

1-d)

Lag ein metode som ut frå ein tekststreng *s* og ein tekststreng-tabell *ordliste* finn ein tekststreng i tabellen med kortast moglege avstand til *s*. Det kan fins fleire tekststrengar med like kort avstand. Metoden kan då fritt velje ein av desse, til dømes den første.

```
public String lignerMest(String s, String[] ordliste)
```

Oppgåve 2 (20%)

-a)

Det er ønskjeleg å leggje modifiserte versjonar av metodane du laga til oppgåve 1 inn i ei modellklasse *Ordliste*. Denne klassa bør ha sjølve ordlista (ein tekststreng-tabell) som ein privat instansvariabel.

Når ein opprettar objekt av klassa *Ordliste* bør ein kunne overføre namn på tekstfila der ordlista er lagra som parameter til konstruktøren. Du kan rekne med at tekstfila inneheld eitt ord på kvar line, og at talet ord er lagra på første line.

Lag klassa *Ordliste* med konstruktør og instansvariablar. Du treng ikkje gjenta koden til metodane frå oppgåve 1, men få fram eventuelle endringar i parametrar og returverdiar.

2-b)

Du skal forklare korleis ein kan programmere ei modellklasse *Orddeler* for å hente ut eitt og eitt ord frå ei tekstfil. Du skal altså ikkje skrive kode, men legg vekt på å få fram kva for instansvariablar du vil nytte, og forklar korleis kvar metode kan programmerast. Klassa skal vere bygd opp slik:

```
public class Orddeler extends Object {

  // Deklarasjon av nødvendige instansvariable

  public Orddeler(String filnavn) {  }
  public boolean flereOrd() {  }
  public String nesteOrd() {  }
}
```

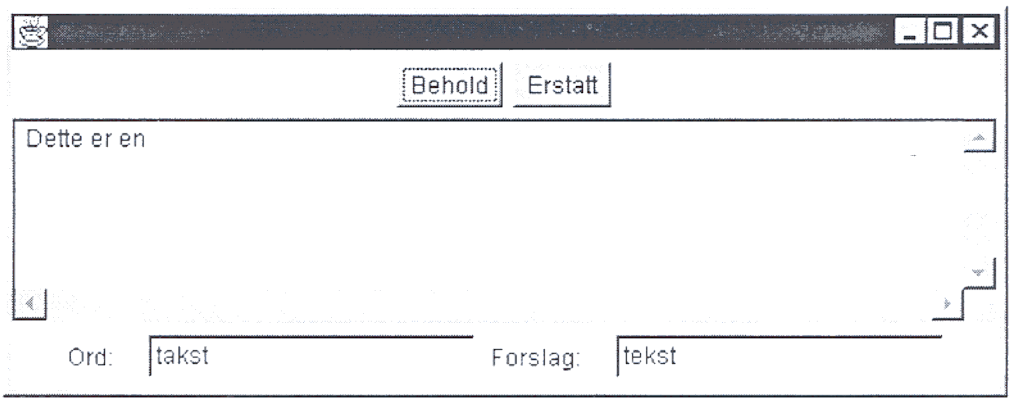
Forklaring til metodane:

- Konstruktøren: Når ein lagar eit objekt av klassa *Orddeler* blir filnamnet overført som parameter. For å forenkle programmeringa reknar vi med at punktum og blank er einaste skileteikn mellom ord.

Metoden flereOrd: Det skal vere mogleg å hente ut eitt og eitt ord frå ein orddelar. Denne metoden skal returnere *true* viss det er fleire ord å hente ut, og elles *false*.
Metoden nesteOrd: Eit kall på denne metoden skal returnere neste ord frå tekstfila. Viss tekstfila inneheld setninga "Dette er ein tekst." skal fire påfølgjande kall på metoden returnere "Dette", "er", "ein" og "tekst".

Oppgåve 3 (40%)

Du skal nå lage ein applikasjon for stavekontroll ved å nytte modellklassane *Ordliste* og *Orddeler*. Brukargrensesnittet skal sjå slik ut:



Du kan rekne med at ordlista er lagra med namn "ordliste.txt" og at fila som skal stavesjekkast er lagra med namn "innfil.txt", begge i same katalog som det køyrbare programmet er lagra. Tekstboksen nede til venstre inneheld eit ord frå innfila, og tekstboksen nede til høgre inneheld til einkvar tid det ordet frå ordlista som liknar mest.

Programmet skal ha følgjande funksjonalitet:

- Ved oppstart skal det opprettast eit *Ordliste*-objekt og eit *Orddeler*-objekt. Då blir óg tekstfilene leste inn.
- Ved oppstart skal programmet óg hente det første ordet frå innfila og det ordet frå ordlista som liknar mest, og syne desse i tekstboksane i nedre del av vindauget.
- Ved klikk i kommandoknappen merkt *Behold* skal ordet frå innfila leggjast til i det store tekstområdet midt i vindauget. Ved klikk i kommandoknappen *Erstatt* skal ordet frå ordlista leggast til i staden. I begge tilfeller skal programmet deretter hente neste ord frå innfila og det ordet som liknar mest frå ordlista, og sette desse inn i tekstboksane for ny handsaming av brukar.
- Når det ikkje er fleire ord å hente frå innfila skal dei to tekstboksane tømmast. Ein kunne då tenke seg at den modifiserte teksten ble lagra til fil, men det skal du ikkje programmere her.
- Ord som er i ordlista skal programmet automatisk leggje til i tekstområdet utan at brukar må klikke *Behold*.

Lag klassa *Stavesjekk* som definerer brukargrensesnittet og kallar på dei riktige metodane i modellklassene *Orddeler* og *Ordliste*.