INF1000 - Uke 11

Ukeoppgaver 11

30. oktober 2012

Vanlige ukesoppgaver

Oppgavene denne uken omhandler de temaene vi har hatt om til nå. Denne uken er det en del oppgaver om det å manipulere tekst på ulike måter, og oppgaver med bruk av metoder, klasser og HashMaps.

Nøkkeloppgaver er oppgaver merket med **[Nøkkeloppg.]** og er plukket ut som spesielt representativ for de viktigste temaene fra ukens forelesning, og alle bør ha som minimumsmål å løse denne selvstendig.

Oppgave 1: Tekster og metoder

Sammenlign to tekster

Skriv en metode `boolean sammenlign(String x, String y)`. Den skal sjekke om `String x` og `String y` er like, og returnere true hvis de er det, ellers false. Du skal gjøre dette uten å bruke Java-biblioteket sine metoder `equals`, `equalsIgnoreCase`, `compareTo` eller `compareToIgnoreCase`.

Oppgave 2: I/O, tekster

Fjern tegn fra tekst

a) Nå skal du skrive et program som leser inn en tekst fra fil, fjerner alle skilletegn (punktum, komma, spørsmålstegn osv) og skriver den nye teksten til en ny fil. Disse tegnene skal fjernes:

```
char[] skilletegn = {'.', ',' ,'?', '!', '"', '(', ')', ':', ';', '/'};
```

Eksempel:

```
Innfil: Hallo! WORLD.Bye, see you later (I think?)
Utfil: Hallo  WORLD Bye  see you later  I think
```

Test programmet ved å lime inn selvvalgt tekst inn i en fil.

b) Det er også en annen måte å gjøre denne oppgaven, vi kan bruke ascii-verdiene til tegnene. Slå opp i en ascii-tabell (f.eks ved å google det), og se at bokstaver har verdiene 65-90 og 97-122. Du kan nå gå igjennom hele teksten, og bytte alle andre tegn med et mellomrom. Husk at du også må teste om tegnene er æ, ø eller å hvis du leser gjennom norske filer.

Oppgave 3: Tekster

Tell forekomster av er ord

Lag et program som tar to argumenter: filnavn og ord. Programmet skal lese av fila fra argumentet og telle hvor mange ganger det gitte ordet forekommer i fila og skrive resultatet ut til skjerm. Eksempel:

```
> java TellOrd olebrumm honning
Ordet "honning" forekom 2 ganger i fila "olebrumm"
```

Oppgave 4: Løkker, String

Tre små kinesere

I den kjente barnesangen tre små kinsere synger man først verset normalt før man bytter ut alle vokalene med a, så e og så i osv. Teksten finner du her: `http://www.barnesanger.no/tre-sma-kinesere.html`.

Skriv et program som lagrer teksten og alle vokalene og så skriver ut hele sangen. Første gang skriver den ut verset slik som det orginalt er, andre gangen byttes alle vokaler ut med en a (Tra sma kanasara ..."), tredje gangen med e osv. Utskriften skal se noe slikt ut:

```
Tre små kinesere på Højbro plass
satt og spilte på en kontrabass.
Så kom en konstabel, spurte hva det var,
tre små kinesere på Høybroplass.

Tra sma kanasara på hajbra plass
satt a spalta pa an kantrabass
sa kam an kanstabel, sparta hva da var
tra sma kanasara på hajbra plass

Tre sme kenesere... osv
```

NB: Utbytting av vokalene kan gjøres veldig enkel ved hjelp av metodene til klassen `String` (se Java-dokumentasjonen), men prøv å skrive hele denne delen selv før du prøver `String` sin metode.

Oppgave 5: [Nøkkeloppg.] Arrayer, tekster

Ordfrekvenser

Tips: For å bruke en hvilken som helst fil her, kan man bruke programmet fra oppgave Fjern tegn fra tekstfor å fjerne tegn. For å telle antall unike ord i fila, kan man skrive kommandoen ipython i terminalen og skrive inn denne linja:

```
print len(set(w.lower() for w in open('eksempelfil').read().split()))
```

Og så kan du sette inn tallet dette gir på første linje i tekst-fila du vil telle ord i.

a) I denne oppgaven skal du lese inn en tekst/artikkel og telle forekomster av de forskjellige ordene. I første linje i fila er det et heltall som sier hvor mange ulike ord det finnes. Du skal opprette et String-array hvor du legger inn de forskjellige ordene, og et int-array av samme lengde hvor du legger inn antall forekomster. `ord[0]` vil da ha `frekvens[0]` forekomster. For hvert ord du leser inn (med unntak av det første), må du sjekke om akkurat det ordet finnes i arrayet allerede. Hvis det ikke gjør det, legger du det inn på neste ledige plass. Hvis det finnes, øker du riktig indeks i frekvens-arrayet med 1. F.eks:

```
if(innlestOrd.equals(ord[i])) {
    frekvens[i]++;
}
```

I tillegg vil vi utelukke tegn (punktum og komma osv) og tall. Det kan også lønne seg å gjøre om alle bokstavene i teksten til små bokstaver, slik at `Det` og `det` blir samme ord. Til slutt kan du sjekke hvilke ord som er de mest frekvente ordene ved å løpe gjennom arrayet på nytt og finne hvilke tall i frekvens-arrayen som er størst. Du vil nå se at ordene det finnes flest av ikke alltid er de mest informative ordene. Hvis du vil kan du endre oppgaven til å bare skrive ut ord som er lengre enn 3 bokstaver og se hva slags ord du nå får.

b) (Klasser og objekter) Gitt klassen `Orddata` (se under), skal du lese inn denne fila, og telle de X ulike ordene som finnes der. Hvert ord skal lagres ved hjelp av klassen `Orddata`, som inneholder ordet og antall forekomster av dette ordet. Øverst i fila er det et heltall som sier hvor mange ord det er. Du skal opprette en array av `Orddata` som har samme lengde som antall ord i fila.

Når du leser inn fila, skal du hver gang du finner et ord sjekke om du har lest dette før. Hvis du ikke har lest dette før, skal du opprette et nytt objekt av klassen `Orddata` som inneholder ordet og antall forekomster (første gang 1). Hvis du har lest det før, skal du øke frekvensen i riktig `Orddata`-objekt med 1. Til slutt skal du sortere arrayet ved hjelp av `Arrays.sort()`. Da er arrayet sortert, og du kan se på de 10 mest frekvente ordene ved å se på indeksene 0-9.

```
class Orddata implements Comparable<Orddata> {
    private String ord;
    private int forekomster;

    public Orddata(String ord) {
        this.ord = ord;
        forekomster = 1;
    }

    public String hentOrd() {
        return ord;
    }

    public int hentForekomster() {
        return forekomster;
    }

    public void nyForekomst() {
        forekomster++;
```

```
    }

    public int compareTo(Orddata x) {
        return x.hentForekomster() - forekomster;
    }
}
```

c) HashMap I denne oppgaven skal vi erstatte arrayet fra forrige oppgave med et HashMap. Legg inn ordene inn i et HashMap der nøklene er `String` ord og verdiene er `Orddata`. For å sortere ordene kan du bruke følgende kode, der du konverterer HashMappet til en array. Så sorterer du arrayet med `Arrays.sort()`:

```
Orddata[] ordarray = ordsamling.values().toArray(new Orddata[0]);
Arrays.sort(ordarray);
```

Så skal du skrive ut de 10 mest frekvente ordene i fila.

Oppgave 6: HashMap, klasser og objekter

Kasino

Oppgaven bygger på oppgaven *Kort og Kortstokk* fra uke 9, så gjør den hvis du ikke har gjort den. Nå skal du utvide med en klasse `Kasino`, som har et `HashMap` av `Kortstokker` og et array med 20 `Dealere`. Tilsammen har kasinoet 100 kortstokker, så i konstruktøren skal `Kasino` opprette 100 kortstokker og legge disse i `HashMap`-en. For å gjøre dette må du lage en `ID` i hver `Kortstokk`. Dette kan være int-variabler fra 0-99. Så må du tildele hver `Dealer` en `Kortstokk`, og da må hver `Kortstokk` som blir tildelt en `Dealer` taes ut av hashmapet. Når kasinoet åpner, begynner hver `Dealer` å dele ut kortene, som igjen kaller metoden bruk i kortstokken. Hvis en dealer har en kortstokk som mangler kort må denne kortstokken byttes ut.

Stjerneoppgaver

Disse oppgavene er litt vanskeligere enn de vanlige ukeoppgaver, først og fremst ment for de som ønsker litt større utfordringer innenfor ukens tema.

Stjerneoppgave 1: I/O, tekster

Setninger

Søk gjennom en tekst etter et gitt ord, og skriv ut alle setningene som ordet forekommer i.

Stjerneoppgave 2: Klasser og objekter

Matteklasse

a) Skriv en klasse MyMath som skal inneholde en del metoder for ulike matematiske operasjoner. Klassen skal inneholde følgende metoder:

```
double add(double a, double b);
double subtract(double a, double b);
double multiply(double a, double b);
double divide(double a, double b);
double abs(double a);
double max(double a, double b);
double min(double a, double b);
double round(double a);
double pow(double a, double b)
```

Alle metodene skal programmeres uten ferdigdefinerte metoder i Javabiblioteket. Du kan gjerne legge til flere metoder du synes er nyttige for en slik klasse å ha.

b) Test programmet ditt ved å la brukeren gi ønsket kommando og tall i terminalen. Du bestemmer selv hvordan brukeren skal kunne gi kommandoer, men husk at brukeren må få informasjon om hvordan han/hun skal bruke programmet.