Oblig 4 (av 4) INF1000, høsten 2009
Værdata, leveres innen 6. nov. kl. 16.00

Formål

Formålet med denne oppgaven er å gi trening i hele pensum og i å lage et større program. Løsningen du lager skal være objektorientert. Det innebærer blant annet at datastrukturen (den delen av programmet som skal "holde på dataene") skal gjøre bruk av objekter av klasser som du selv definerer. ***Oppgaven skal besvares individuelt – enhver kopi av hele eller deler av andres løsninger vil bli behandlet som fusk***! For mer detaljert informasjon om hvordan du kan gå fram for å løse oppgaven, se avsnittet *Tips & krav til løsningen.*

Oppgave

Meteorologisk institutt har en rekke værstasjoner rundt i Norge. Ved hver stasjon gjøres det daglig en rekke værmålinger, og resultatene samles på store datafiler for senere analyser. Hensikten med denne oppgaven er å lage et program som kan lese slike meteorologiske data fra fil, legge dataene inn i en fornuftig datastruktur, gjøre noen relativt enkle beregninger og skrive ut resultatene på skjerm. Programmet skal være kommandostyrt.

Du skal i denne oppgaven bruke data som du finner på to filer: og "**`KlimaData-v2009.txt`**" og "**`StasjonerOgDataBeskrivelser-v2009.txt`**" , som du finner og laster ned fra hjemmesida til INF1000:
`http://www.uio.no/studier/emner/matnat/ifi/INF1000/h09/`

Ser vi først på fila: "**`StasjonerOgDataBeskrivelser-v2009.txt`**", ser vi at den i linje 2 til 27 har opplysninger om 26 værstasjoner i Norge, og av overskrifta i linje 1 ser vi at dette er: Et entydig stasjonsnummer, stasjonens navn, stasjonens høyde over havet, kommune, fylke og landsdel.

Alle data på disse 26 linjene skal du lese inn i programmet ditt (Hint, bruk: **`inLine()`** til å hoppe over første linje). De øvrige linjene 28-46 er en beskrivelse av hva du finner på den andre fila ("**`KlimaData-v2009.txt`**") og skal bare leses av deg, f.eks i en editor.

Fila " "**`KlimaData-v2009.txt`**" inneholder observasjoner fra alle dagene i jan.-juni 2009 for de 26 værstasjonene (den første linja fra Klima-data fila er også kopiert inn på linje 46 i "**`StasjonerOgDataBeskrivelser-v2009.txt`**" , slik at du kan se hvilke data som er registrert og i hvilken rekkefølge) **N.B.** Mangler noen av observasjonene er det markert med verdien -999. Programmet ditt må sørge for å ikke ta med denne -999 verdien når det sammenlignes verdier eller regnes ut gjennomsnitt. Den første linja betyr da:

```
180     01.01.2009     0,6   1,1   0,1   -999  -20,4  -24,5  -16,2
```

at det på stasjon 180 (TRYSIL_VEGSTASJON), var det den 01.01.2009 en midlere vindhastighet på: 0,6 m per sekund, høyeste vindhastighet: 1,1 meter per sekund, nedbøren var: 0,1 mm; det mangler måling av snødybden (-999), middeltemperaturen var: -20,4 °C, minimum temp: -24,5°C og maksimum temp: -16,2.

Oppgave 1)

Hele denne oppgaven skal løses med klasser og objekter. Ut fra opplysningene ovenfor og ved å lese gjennom de spørsmål systemet skal kunne gi svaret på i oppgavene 2-7, så skal du tegne et

klassediagram i UML over systemet ditt. Gi navn på de forhold du vil ha i systemet ditt. Skriv også antall på begge sidene av forholdene. Hvis du greier å få skannet inn denne tegningen, kan du levere den inn sammen med det andre du lager, hvis ikke, skal du kunne levere den inn hvis hjelpelæreren ber om det.

Oppgave 2)
Du skal nå skrive de delene av programmet som leser de to filene, først linjene 2 til 27 fra "**`StasjonerOgDataBeskrivelser-v2009.txt`**" og deretter alt fra "**`KlimaData-v2009.txt`**". Du oppretter objekter av de klassene du har definert etter hvert som du leser filene. Alle klassene (med unntak av den som inneholder **`'main'`**) skal ha en konstruktør du har skrevet selv og som initierer objektet med de data du leser inn om vedkommende objekt. Navnene på disse filene skal være parametere når du starter programmet (dvs. filnavnene er i **`args[0]`** og **`args[1]`** som parametere til **`main`**, se hint).

Oppgave 3)
Du skal nå utstyre programmet med en metode som skriver ut en meny for brukeren og som gir brukeren følgende valg, som du skal programmere i oppgavene 4-6:

```
1 – finnAntallUværsDager(int stasjonsNummer, int måned)
2 – sammenlignRegnVedKyststasjonerMotResten()
3 – sammenlignØstlandetMotNordNorge(int måned)
4 – finnKaldesteVærstasjonMåned(int måned)
5 – avslutt
```

Oppgave 4)
Du skal skrive metoden :

```
finnAntallUværsDager(int stasjonsNummer, int måned)
```

som går gjennom alle observasjonsdata for den brukervalgte værstasjonen og for den måned brukeren også har valgt, og summerer antall døgn hvor ***summen av*** nedbør i mm og maksimal vindhastighet i m/sek er større enn 10.7 (dvs. enten blåser det mye eller det regner mye – eller begge deler. Tallet 10.7 er valgt fordi hvis det ikke regner den dagen, så er det minst kuling i vindstyrken). Skriv ut stasjonsnavn, måned og antall dager med uvær du fant for denne stasjonen i den valgte måneden.

Oppgave 5)
Du skal skrive metoden :

```
sammenlignRegnVedKyststasjonerMotResten()
```

som prøver å belyse en påstand at det regner mer på værstasjoner som ligger ved kysten enn ved de andre stasjonene (som da ligger inne i landet). Vi skal definere at alle værstasjoner som ligger mindre enn 60 meter over havet, ligger ved kysten og resten ligger i innlandet. Gå gjennom alle data du har registrert og lag gjennomsnitt mm nedbør for kyststasjonene per dag (for alle de månedene du har data for) og skriv dette gjennomsnittet ut sammen med tilsvarende gjennomsnitt for "innlandsstasjonene". Skriv så ut en kommentar fra programmet som sier om påstandene om mer regn på kysten er riktig.

Oppgave 6)
Du skal skrive metoden :

`sammenlignØstlandetMotNordnorge(int måned)`

som i undersøker påstanden at det i gjennomsnitt regner mindre ***og*** samtidig er varmere på Østlandet enn i Nordnorge i den valgte måneden.

Du regner så ut og skriver ut på en egen fil "**`Resultat.txt`**" for den valgte måneden gjennomsnittstemperatur, samt gjennomsnittlig nedbør for de to gruppene av værstasjoner (nordnorske og østlandske). Til slutt skriver programmert ditt ut en kommentar om påstanden ovenfor var riktig.

Oppgave 7)
Skriv metoden

`finnKaldesteVærstasjonMåned(int måned)`

som går gjennom data for alle stasjonene for den gitte måned og skriver ut navnet på den stasjonen hvor *gjennomsnittet av minimumstemperaturene* for hver dag den måneden er minst for samtlige stasjoner. Du skal også skrive hvilken måned som var kaldest for den stasjonen.

Levering av følgende for Oblig-4 innen . nov kl. 16.00:
Du skal levere via Joly, som bare tar to filer. Hvis du *ikke* greier å lage en .zip fil så leverer du:

1. Programmet (.java filen)
2. En av html-filene som lages av Javadoc av en av klassene dine (husk å legge inn javadoc-kommentarer)

Dersom du greier å lage en .zip fil, så skal du alltid som første fil til Joly sende med .java filen (programmet) og .zip fila skal være den andre fila til Joly og skal da inneholde:

3. Den filen "**`Resultat.txt`**" du laget i oppgave 6.
4. En av html-filene som lages av Javadoc av en av klassene dine
5. UML klassediagrammet ditt (dersom du greier å skanne det inn som en fil – eller at du har laget det med et tegneverktøy).

Tips & krav til løsningen.:

a) Et viktig valg er hvilke klasser du har i programmet. I 'mønstreløsningen' er det følgende klasser**: Oblig4, MetInst, Stasjon, Maaned, Dag .**

Slik begynner mønsterløsningen (med Javadoc kommentarer):

```
import easyIO.*;

/**
* Omsluttende klasse for problemet, tar opp parametre
* fra kommandolinja og starter kommandoløkka. Feilmelding
* hvis ikke minst to parametere.
*******************************************************/
class Oblig4 {
        /**
        * Sjekker parametere, starter opp ordreløkka etter at
        * filene er lest via konstruktoren til 'MetInst'
        *******************************************************/
    public static void main(String[] args) {

        if (args.length >= 2) {
```

```
                MetInst m = new MetInst(args[0],args[1]);
                m.ordreløkke();
        } else
                System.out.println("Bruk: >java Oblig4 <fil med Stasjonsdata> < fil med Observasjonsdata>");
    }
} // end class Oblig 4
```

b) Lag tre hjelpemetoder i den klassen som representerer måledata for en måned for en stasjon. Den første regner ut gjennomsnittlig nedbør per dag den måneden; den andre metoden regner ut gjennomsnittlig temperatur for måneden ut fra middeltemperaturene for hver dag, og den siste metoden teller opp antall uværsdøgn..

c) Av og til ønsker du å søke etter stasjonene ut fra deres stasjonsnummer og av og til ut fra deres navn. Det kan derfor være fornuftig å lage *to* HashMap'er – en hvor nøkkelen er navnet og en hvor nøkkelen er stasjonsnummeret. Det er det *samme* objektet du legger (peker til) i de to HashMap'ene, men nøklene vil være ulike.

d) Hvis du vil lage en nøkkel til en HashMap av noe som egentlig er et heltall, si: `int num;,` så lager du enkelt en nøkkel slik:

```
String  s = num+"";
```

(dette lager en String med tallverdien i num ved å legge til den tomme tekststrengen)

e) En måte å behandle manglende data, er at alle -999'ene leses inn som om dette er virkelige data. De metoder som regner ut gjennomsnitt ol, må da hver gang sjekke om det er verdier != -999 de finner og ikke ta med slike dataverdier i beregningene. Slike metoder kan da også selv returnere -999 dersom det ikke finnes **noen** data å gjøre beregningene på (enten f.eks. for at brukeren spesifiserer en måned vi ikke har data for - f.eks. måned: 8 – august, eller at **alle** data for vedkommende måned mangler). Husk at du samtidig må telle opp hvor mange dager du har virkelige observasjoner for, slik at gjennomsnittet blir riktig.

f) For å få de to filnavne som parametre til 'main':
så nevner du dem bare på samme linje som du når du starter programmet ditt. Anta at klassen din som inneholder main heter 'Meteorologi' . Du starter da programmet som følger:

```
>java Meteorologi StasjonerOgDataBeskrivelser-v2009.txt   KlimaData-v2009.txt
```

g) Hvis du leser inn data og så skriver dem ut på en PC via Windows, så vær klar over at æ, ø, å, Æ, Ø, Å blir skrevet ut som andre tegn på skjermen, og hvis man for eksempel taster inn æ, ø, å på en Windows-pc, så vil de ikke bli oppfattet som de æ-ene, ø-ene og å-ene du har lest inn fra fila. *Bruk derfor stasjonsnummerene* til å identifisere stasjonene, både i brukerdialogen og innad i programmet når du leser Vaerdata-filen. (eksempler på 'mishandling' av ÆØÅ er:TORSV┼G_FYR(90800) MIDTL╞GER (46510),). Skriv derfor både ut navn og nummer på stasjonene når bruker skal velge stasjon. Da kan brukeren taste inn et stasjonsnummer når stasjonen velges..

h) Hvis du får behov for å lage gjennomsnitt av f.eks nedbør fra 2 eller flere måneder som har ett ulikt antall dager i seg med virkelige observasjoner, så ikke lag noe problem ut av det. La slike data for alle månedene telle likt i et større gjennomsnitt (del bare på antall måneder).

i) Hvis du har lyst til bare å nytte engelske variabelnavn, metodenavn og dokumentasjon, så gjør gjerne det, men brukerdialogen skal være på norsk.

j) Når du skal lage javadoc av systemet ditt (og vi antar at de eneste .java – filene på det filområdet er de som hører til Oblig4), så gir du kommandoen:

```
>javadoc –package *.java
```

Grunnen til å å ha med parameteren **`–package`** er at den gjør at alle variable, klasser og metoder som det ikke står noen modifikator foran samt de det står public foran blir med i dokumentasjonen. Har du ikke med **`–package,`** vil bare de som er nevn som **`public`** blir med i dokumentasjonen.

Husk også at du kan lage korte javadoc-kommentarer på en linje som f.eks:

```
/** Skriver ut bruker-menyvalg */
 void  meny (Out ut) {
...
```

Javadoc-kommentarer for en objektvariabel plasseres like over deklarasjonen, som f.eks:

```
/** Stasjonenes høyde over havet (meter) */
int moh;
```

k) Når du lager filen ”resultat.txt” i oppgave 6, kan det være fornuftig å åpne den med append, dvs. f.eks.:

```
Out res = new Out("Resultat.txt",true);
```

Gjør du det, vil alle resultatene fra testingen av Nordnorge mot Østlandet for alle månedene du prøver bli lagret på fila (du overskriver da ikke gamle data, men skriver nye resultater på slutten av fila).

l) Mønsterløsningen ble på ca. 420 linjer java-kode + 70 linjer med javadoc-kommentarer, men din løsning kan godt bli betydelig kortere eller lengre uten at den er noe dårligere for det.

Lykke til !