Oblig 4 – Værdata, leveres 14 november kl. 16.00

Du får gitt følgende problemstilling som du skal lage et datasystem for.

Meteorologisk institutt har en rekke værstasjoner rundt om i Norge. For hver slik værstasjon får vi oppgitt et entydig stasjonsnummer, stasjonens navn, stasjonens høyde over havet, kommunen og fylket hvor stasjonen ligger. En oversikt over de stasjonene vi først skal behandle data fra, finnes på filen "**`Stasjoner-1.txt`**". Det er data for én stasjon på hver linje, og opplysningene er atskilt med en eller flere blanke. Den følgende linjen er data for værstasjonen på Gardermoen flyplass.

```
4780  GARDERMOEN    202  ULLENSAKER        AKERSHUS
```

Ved disse værstasjonene måles hver dag en rekke data, og vi skal her konsentrere oss om følgende fire data per dag: Maks vindhastighet, mmNedbør, Min temp og Max temp. Data for én måned for en værstasjon samles og rapporteres ofte som en enhet. På filen "**`Vaerdata-1.txt`**" finner du målinger for de seks første månedene i 2003 for de værstasjonene som beskrives av "**`Stasjoner-1.txt`** "ovenfor. Hver linje i denne filen med måledata inneholder følgende opplysninger for én dag: stasjonsnummer, dag-i-måned, måned, max vindhastighet (m/sek), mm nedbør, min temperatur og max temperatur. Følgende to linjer er data for Gardermoen værstasjon for den 14. og 15. januar 2003:

```
4780 14 01 10.8  0.3  -2.4   6.2
4780 15 01  8.7  0.0   1.4   5.0
```

(av den første linjen kan vi da lese at på GARDERMOEN værstasjon den 14 januar blåste det 10.8 m/sek i maks vindhastighet, det regnet 0.3 mm nedbør og min temp var -2.4°C og max. temp var 6.2°C det døgnet)

N.B. For noen av datafeltene mangler observasjoner, og da står dataverdien -99 der det skulle stått en observasjon (typisk gjelder dette ofte max vindhastighet). Programmet ditt må sørge for å ikke ta med denne -99 verdien når det sammenlignes verdier eller regnes ut gjennomsnitt!

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 2-6, 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. Lever gjerne UML-diagrammet som en håndtegning.

Oppgave 2)
Du skal nå skrive de delene av programmet som leser de to filene "**`Stasjoner-1.txt`**" og "**`Vaerdata-1.txt`**". Det oppretter objekter av de klassene du har definert etter hvert som du leser filenes. Alle klassene (med unntak 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 som gir brukeren følgende valg, som du skal programmere i oppgavene 4-6 (Menyen skal se ut nøyaktig som nedenfor):

```
1 – Finn antall uværsdager
2 – Sammenlign regn ved kyststasjoner mot resten
3 – Sammenlign landsdeler
4 – avslutt
```

For hvert menyvalg skal det kalles en metode, som beskrives i oppgave 4-6. Når du har fått oppgavene 1-3 til å virke (uten at du hittil har skrevet kode for metodene som beskrives nedenfor), kan du velge å gå over til å bruke filene "**`Stasjoner-2.txt`**" og "**`Vaerdata-2.txt`**" som inneholder data for langt flere stasjoner, og som derfor kan gi bedre og mer interessante svar enn de få værstasjonene det er i de tilsvarende filene som slutter med "**`-1.txt`**".

Oppgave 4)
Dersom brukeren velger menyvalg 1 skal brukeren blir bedt om å taste inn stasjonsnavn og måned. Det skal se slik ut:

```
Skriv stasjonsnavn: Gardermoen
Skriv måned: 11 |
```

Du skal skrive metoden som så blir kalt:

```
finnAntallUværsDager(String stasjonsnavn, int måned)
```

Metoden 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 (på skjermen) stasjonsnavn, måned og antall dager med uvær du fant for denne stasjonen i den valgte måneden.

Oppgave 5)
Du skal skrive metoden som kalles når brukeren velger menyvalg 2:

```
sammenlignRegnVedKyststasjonerMotResten()
```

Metoden 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 som sier om påstandene om mer regn på kysten er riktig.

Oppgave 6)
Dersom brukeren velger menyvalg 3 skal brukeren blir bedt om å taste inn en måned på denne måten:
`Skriv måned: 11 |`
Du skal skrive metoden som så kalles:

```
sammenlignLandsdeler(int måned)
```

Den undersøker hvordan nedbør og temperatur varierer mellom landsdelene i den valgte måneden. Gjennomsnittstemperatur for en dag sier vi er: (maxtemp + mintemp)/2.0 for den dagen.

Før du går løs på denne oppgaven, innfører du en char-variabel: **`landsdel`** i klassen som representerer en værstasjon. Den skal ha verdien 'Ø' hvis fylket ligger på Østlandet, 'V' hvis fylket ligger på Vestlandet, 'S' hvis det ligger på Sørlandet, 'M' hvis det ligger i Midt-Norge og 'N' hvis fylket ligger i Nord-Norge.

Hvilken landsdel et fylke ligger i skal leses inn fra filen **`fylker.txt`**. Filen inneholder alle fylkene i landet med en av de nevnte bokstavene bak. Før programmet leser de to tidligere nevnte filene med stasjoner og måliger leses denne filen og alle fylkene legges i en HashMap. Indeks skal være navnet på fylket og verdien skal være en av de nevnte bokstavene for landsdeler.

Du regner så ut gjennomsnittstemperatur (definert som snittet av max- og min-.temperaturene) og gjennomsnittlig nedbør for de forskjellige landsdelen og skriver alle verdiene ut på en egen fil "**`Resultat.txt`**" for den valgte måneden. Linjene i filen skal være på formen nedenfor.

```
Østlandet i måned:  11  Temperatur:  7.0,  Nedbør: 0.3
Vestlandet i måned: 11  Temperatur: 11.2,  Nedbør: 0.5
Sørlandet i måned:  11  Temperatur:  7.6,  Nedbør: 0.2
Midt-Norge i måned: 11  Temperatur:  7.6,  Nedbør: 0.2
Nord-Norge i måned: 11  Temperatur: 12.2,  Nedbør: 0.3
```

I den oppgaven skal du legge til nye linjer i fila hver gang og ikke skrive over tidligere resultater.

Levering skal gjøres via Joly innen 14. nov kl. 16.00.

URL: http://obelix.ifi.uio.no:8080/wizard.html
Du skal levere følgende:

- Klassediagrammet
- En av html-filene som lages av Javadoc for en av klassene dine (husk å legge inn javadoc-kommentarer)
- Selve koden (alle klassene)
- Utskrift av filen "**`Resultat.txt`**" du laget i oppgave 6.

Her student skal levere hver sin egenprogrammerte løsning, og du plikter å ha lest og forstått følgende krav til innleverte oppgaver ved institutt for informatikk:
http://www.ifi.uio.no/studinf/skjemaer/erklaring.pdf

Når du leverer oppgaven elektronisk i Jolysystemet må du legge en kommentar som første linje i .java-filen din som sier hvem du er, login-navnet, gruppe, oblignummer, kurskode og semester som flg:

```
//<brukernavn> g-<gruppenummer> o-<oblignummer> k-<kurskode>/<semester>.
```

Eks for Arne Maus (arnem) på gruppe 12:

```
// arnem g-12 o-2 k-inf1000/h07 ).
```

Se også kursets hjemmesider for oppdateringer og beskjeder.

Noen hint og kommentarer til oppgaven:

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:

```
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 max- og min-temperaturene 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 `-99`'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 `!= -99` de finner og ikke ta med slike dataverdier i beregningene. Slike metoder kan da også selv returnere `-99` 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) De to filnavnene som parametere til "main"
For å få til det så nevner du dem bare på samme linje som du når du starter programmet ditt. Anta at klassen din som inneholder "main" i en klasse som heter 'Meteorologi'. Du starter da programmet som følger:

```
>java Meteorologi Stasjoner-1.txt Vaerdata-1.txt
```

Når du senere vil prøve de større filene med flere måledata, sier du bare:

```
>java Meteorologi Stasjoner-2.txt Vaerdata-2.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 f.eks 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 stasjonsnumrene* til å identifisere stasjonene, både i brukerdialogen og innad i programmet når du leser Vaerdata-filene. (eksempler på "mishandling" av ÆØÅ: NY-┼LESUND (99910), MIDTL├GER (46510), VARD╪ (98550)). Skriv derfor både ut navn og nummer på stasjonene. Hvis du har dette problemet kan du be brukeren taste inn et stasjonsnummer når stasjonen velges, men du skal likevel kalle f.eks. finnAntallUværsDager(..) med den tilhørende teksten som er stasjonsnavnet.

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 benytte 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, bør du åpne den med append, dvs. f.eks.:

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

Gjør du det, vil alle resultatene fra testingen 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!