Oblig 4 – Værdata, leveres 11. november kl. 16.00

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

Meteorologisk institutt har en rekke værstasjoner rundt i Norge. For hver slik værstasjon har vi oppgitt et entydig stasjonsnummer, stasjonens navn, stasjonens høyde over havet, kommune og fylke hvor stasjonen er. 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), , mintemp, maxtemp og mm-nedbør. Følgende to linjer er følgelig data for Gardermoen værstasjon for den 14. og 15. januar 2003:

```
4780 14 1  10.8  -2.4   6.2   0.3
4780 15 1   8.7   1.4   5.0   0.0
```

(av den første linjen kan vi da lese at på GARDERMOEN værstasjon den 14 jan blåste det 10.8 m/sek i maks vindhastighet, og min temp var -2.4 C, max. temp var 6.2 C og at det regnet 0.3 mm nedbør 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. (Denne kan leveres direkte til din gruppelærer eller i luken til ekspedisjonskontoret på ifi)

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:

```
1 – finnAntallUværsDager(String stasjonsnavn, int måned)
2 – sammenlignRegnVedKyststasjonerMotHøyder()
3 – finnSeksVarmesteStasjoner(int måned)
4 – avslutt
```

Når du har fått Oppgavene 1-3 å virke (uten at du har hittil har skrevet kode for metodene), 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.

Oppgave 4)
Du skal skrive metoden :

`finnAntallUværsDager(String stasjonsnavn, 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. Test for minst 2 måneder og 2 stasjoner – fritt valgt (dvs. minst 4 tester).

Oppgave 5)
Du skal skrive metoden :

`sammenlignRegnVedKyststasjonerMotHøyder()`

som prøver å belyse en påstand at det regner mer på værstasjoner som ligger ved kysten enn ved stasjonene som ligger i høyden. Vi skal definere at alle værstasjoner som ligger mindre enn 60 meter over havet, ligger ved kysten og de over 100 meter som i høyden. 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 "høydestasjonene". Skriv så ut en kommentar som sier om påstandene om mer regn på kysten er riktig.

Før du går løs på denne oppgaven, innfører du to boolske variable: **`kystStasjon`** og **`stasjonIHøyden`** i klassen som representerer en værstasjon. Den første, **`kystStasjon`**, settes lik **`true`** i konstruktøren hvis stasjonen er plassert mindre enn 60 meter over havet, og **`stasjonIHøyden`** settes **`true`** hvis værstasjonen ligger plasser mer enn 100 meter over havet. (En del stasjoner vil selvsagt ha **`false`** på begge disse variablene).

Oppgave 6)
Du skal skrive metoden:

`finnSeksVarmesteStasjoner(int måned)`

Denne skal lage en sortert liste over de 6 varmeste stasjonene i løpet av en gitt måned. Varmest er her definert som den/de dagen(e) med høyest mulig maxTemp.

Programmet skal skrive ut en ferdig sortert liste over disse 6 stasjonene – sortert etter synkende maksTemp (ala 26.6, 24.5, 22.6, 22.0, 21.7, 21.5). Den skal selvsagt også skrive ut hvilken stasjon det er snakk om (Dette kan du enten løse ved hjelp av 2 arrayer (en som har oversikt over temperaturen og en som har stasjonsnavnet), eller ved hjelp av å opprette egne objekter som du lagrer i en 6 plasser lang objektarray.) Du bør uansett lage en variabel som til enhver tid forteller hva som er verdien i den 6.te høyeste temperaturen.

Oppgaven skal løses ***uten*** ferdiglagede sorteringsmekanismer fra javabibliotekene (som for eksempel Arrays.sort). Test for minst 3 måneder.

Du regner så ut og skriver ut på en egen fil "**`Resultat.txt`**" for den valgte måneden.

Levering av følgende for Oblig-4 innen 11. nov kl. 16.00:
Du skal levere klassediagrammet, Javadoc av klassene dine (husk å legge inn javadoc-kommentarer), selve koden og filen "**`Resultat.txt`**" du laget i oppgave 6.
Du skal også legge ved resultatet av en testkjøring av programmet ditt (på samme måte som i oblig 2 og 3).

Noen hint og kommentarer til oppgaven:

a) Lag en boolsk hjelpemetode i klassen du bruker til å lagre dagdata som returnerer true hvis det er uvær eller false ellers. Lag så en hjelpemetode i månedsdataklassen din som summerer antall uværsdager for sin måned.

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..

b) 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.

c) 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)
(Nb: Dette er ikke strengt nødvendig i java 1.5, pga Javas nye "innpakking av enkle variable til objekter" (se side 156 i læreboken), men det bør gjøres både for forståelsens og for feilsøkings skyld.

d) En måte å behandle manglende data, er at alle -99'ene leses inn som om dette er virkelige data, og at metoder som regner ut gjennomsnitt ol, hver gang sjekker om det er verdier != -99 de finner og bare tar med dataverdier som != -99. Slike metoder kan 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.

e) 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 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
```

f) 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 stasjonsnummerene* 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 når bruker skal velge stasjon.

g) 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).

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

i) 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;
```

j) 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 for alle månedene du prøver bli lagret på fila (du overskriver da ikke gamle data, men skriver nye resultater på slutten av fila).

Lykke til!