UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Oppgaver basert på eksamen i Inf1050, 2007

Innledning

Regjeringen i Ruritanien har forpliktet seg til at landet skal redusere utslippet av klimagasser. Regjeringen innser imidlertid at for å få alle innbyggerne til å bidra, er det ikke tilstrekkelig bare med holdningskampanjer. Derfor ønsker myndighetene å registrere alle personlige innkjøp som vil bidra til økt utslipp, fra bensin og fyringsolje til flyreiser. Dersom en persons utslippsbidrag i løpet av et år overstiger et visst nivå, er det planen å ilegge vedkommende en klimaskatt som myndighetene kan bruke til å kjøpe klimakvoter i utlandet.

Heldigvis har Ruritanien allerede innført et elektronisk borgerkort som innbyggerne kan bruke i alle sammenhenger der det kreves en sikker identifikasjon, som for eksempel ved bruk av nettbank, signering av offentlige dokumenter og ved elektroniske valg. Planen er å bruke dette kortet som identifikasjon også ved innkjøp av varer og tjenester som vil føre til økt utslipp av klimagasser. Hver enkelt innbygger skal til enhver tid kunne logge seg inn på www.miapagina.rt for å følge med på innkjøpene og dermed sitt personlige bidrag til utslippene.

Mulige betenkelige sider ved disse planene, som for eksempel svartebørshandel med frikvotene og kjøp av flybilletter i utlandet, skal vi overlate til Ruritaniens politikere å diskutere. Vi skal utelukkende se på noen problemstillinger rundt IT-løsningene.

Oppgave 1 (20 % – 36 min.)

Hvor mye av de ulike klimagasser som slippes ut ved forbruk av et produkt, fremgår av tabellen klimagassmengde_per_produktenhet. For å unngå vanskeligheter med ulike benevninger, er det forutsatt at et gitt produkt alltid måles i samme avtalte, underforståtte enhet (eksempelvis måles bilbensin i liter og flyreiser i kilometer), og at klimagassmengde_per_produktenhet på tilsvarende måte alltid oppgis i samme avtalte, underforståtte enhet (eksempelvis måles CO_2-utslippet fra bilbensin i kilogram/liter). I klimaregnskaper regnes ofte utslipp av alle gasser om til CO_2-ekvivalenter ut fra hvor mye skade de gjør. CO_2 har selvsagt CO_2-ekvivalenten 1, mens CH_4 (metan) har CO_2-ekvivalenten 21.

Her er vist to av de tabellene med forekomster. NULL i en kolonne betyr at verdien foreløpig ikke er kjent.

Klimagass

klimagass	CO2ekvivalent
CO_2	1
CH_4	21
N_2O	310
SF_6	23900
C_2F_6	9200
CF_4	6500

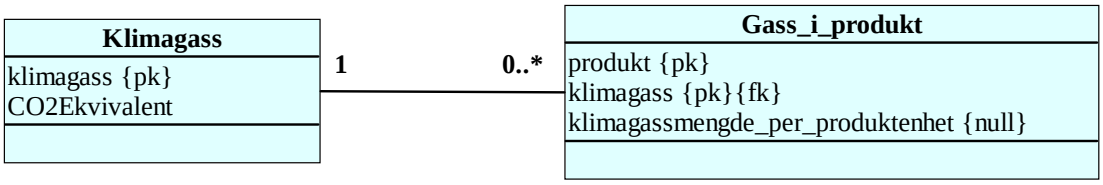
Gass_i_produkt

produkt	klimagass	klimagassmengde_per_produktenhet
bilbensin	CO_2	2,50
bilbensin	CH_4	0,0002
bilbensin	N_2O	0,0005
autodiesel	CO_2	NULL
fyringsolje	CO_2	2,60
flyreise	CO_2	0,17

a) Regn ut hvor mye klimagass bilbensin slipper ut i form av CO2-ekvivalenter pr. produktenhet?

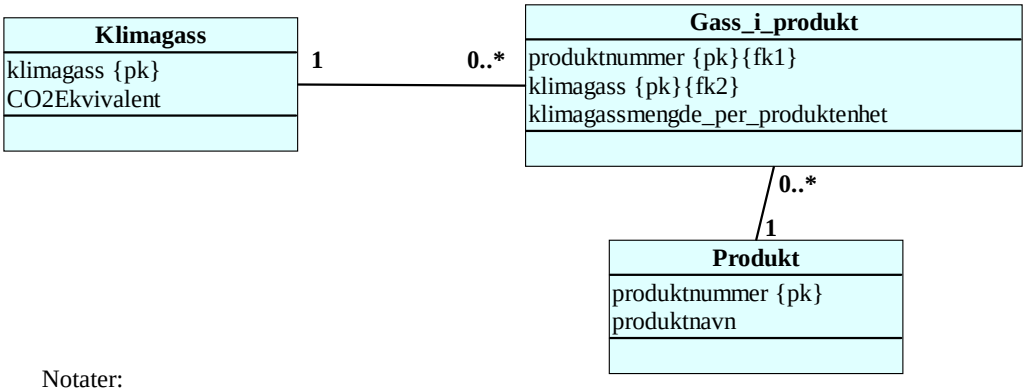
```
2.5*1+0.0002*21+0.0005*310
```

b) Lag et dataorientert UML klassediagram basert på de to tabellene over, som definerer eventuelle assosiasjoner, primærnøkler og fremmednøkler.



Notater:
Primærnøkkelen for Gass_i_produkt er sammensatt av produkt og klimagass (begge har {pk})
klimagass-attributtet er dessuten fremmednøkkel mot tabellen Klimagass
Hvis ikke annet er angitt antas {not null} på alle attributter

c) Lag et dataorientert UML klassediagram hvor man i tillegg til klassene fra oppgave b) innfører en egen klasse som heter Produkt, og som har attributtene produktnummer og produktnavn (for eksempel ”bensin”). Oppdater eventuelle attributter, assosiasjoner, primærnøkler og fremmednøkler i alle klassene slik at diagrammet blir konsistent.



Notater:
Primærnøkkelen for Gass_i_produkt er sammensatt av produktnummer og klimagass (begge har {pk})
klimagass-attributtet er fremmednøkkel mot tabellen Klimagass
produktnummer-attributtet er fremmednøkkel mot tabellen Produkt
Nå er det ikke lenger nødvendig å ha "null" på klimagassmengde_per_produktenhet, siden produktene er definert i en egen tabell.
Hvis ikke annet er angitt antas {not null} på alle attributter

d) Diskuter fordeler og ulemper med de to alternative klassediagrammene i oppgave 1b og 1c.

```
1c vs 1b:
- Mindre dobbeltlagring av produktrelatert info
- bedre utvidbarhet dersom man ønsker å definere flere egenskaper
ved produkter
- Lettere å sjekke at alle produkter i Gass_i_produkt er lovlige
- trenger ikke lenger å tillate NULL i attributtet
klimagassmengde_per_produktenhet
- Flere tabeller => mer kompleks SQL-kode for oppdateringer
```

Oppgave 2 (30 % – 54 min.)

I denne oppgaven skal du fullføre en objektorientert utforming (design) av et bruksmønster som heter ”Beregn Mengde”. Bruksmønsteret starter ved at personen oppgir sitt fødselsnummer. Deretter viser systemet den totale klimagassmengden som personen gjennom sine kjøp har bidratt til, det vil si en totalsum summert over alle klimagasser (omregnet til CO_2ekvivalenter) fra årsskiftet til dags dato for den gitte personen. NB! Følgende tre deloppgaver kan med fordel løses i parallell.

a) Lag en tekstlig spesifikasjon av bruksmønsteret ”Beregn Mengde”. Spesifikasjonen skal inneholde navn, aktør, trigger, normal hendelsesflyt og variasjoner. Sørg for at bruksmønsterspesifikasjonen blir konsistent med sekvensdiagrammet som du skal lage i oppgave 2b).

Navn: Beregn Mengde

Aktør: Person (eller forbruker el.l.)

Trigger: Person ønsker oversikt over sitt totalforbruk

Normal Hendelsesflyt:

1. Personen oppgir sitt fødselsnummer
2. Systemet finner personen i systemet
3. Systemet beregner den totale klimagassmengden som personen gjennom sine kjøp har bidratt til
4. Systemet viser klimagassmengden til personen

Variasjoner:

2a. Ugyldig fødselsnummer:

1. Systemet gir feilmelding og avslutter

2b. Personen finnes ikke i systemet:

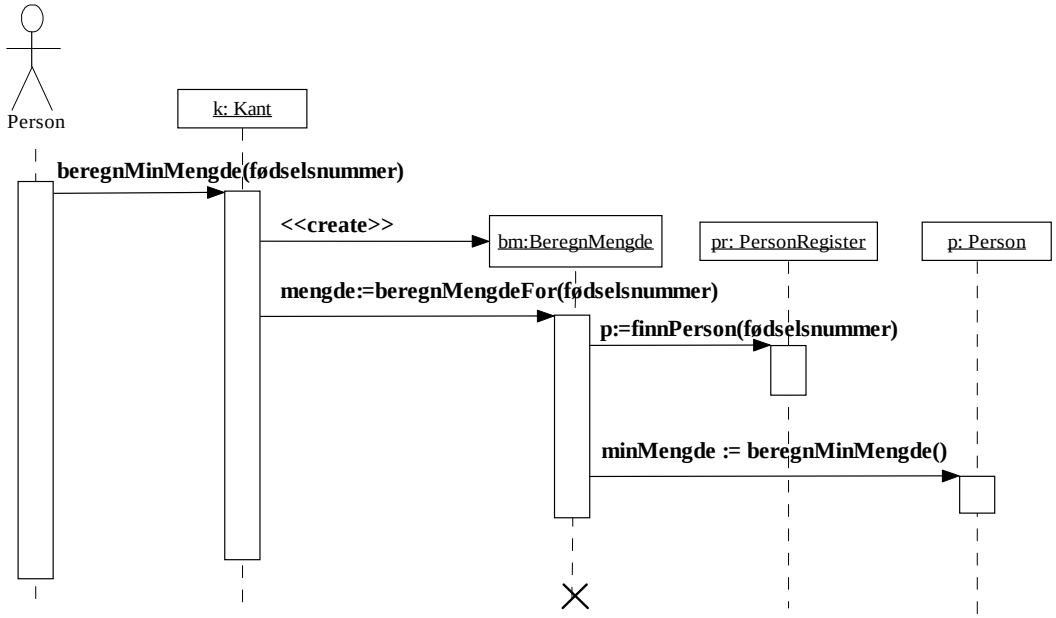
1. Systemet gir feilmelding og avslutter

Kommentarer: Her finnes det selvsagt mange varianter som er greie løsninger. Man kan godt si at hvis man oppgir et ugyldig fødselsnummer så vil man heller ikke finne personen, så hvis man ikke har med noe i likhet med 2a så er det også ok… Men det skal gi trekk dersom bruksmønsteret uttrykker noe helt annet enn sekvensdiagrammet (de skal være konsistente).

b) Lag et sekvensdiagram som tilsvarer normal hendelsesflyt for bruksmønsteret "Beregn Mengde". Sekvensdiagrammet skal inneholde en aktør og nøyaktig fire objekter:
(1) *k:Kant*, (2) *bm:BeregnMengde*, (3) *pr:PersonRegister* og (4) *p:Person*.

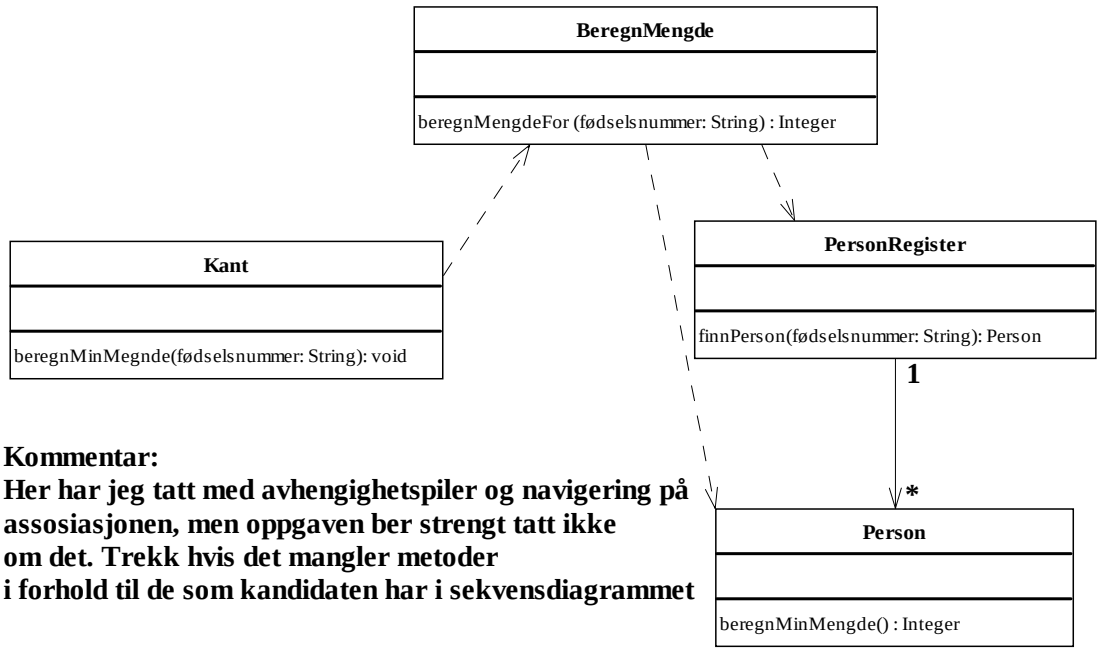
Klassen *PersonRegister* har kun en metode, *finnPerson(fødselsnummer:String): Person*, som returnerer det bestemte personobjektet *p* som har et gitt *fødselsnummer*. Du skal ikke spesifisere *hvordan* denne metoden er implementert, men kan anta at den returnerer riktig personobjekt til kontrollobjektet. Klassen *Person* har kun en metode, *beregnMinMengde():Integer*, som returnerer den totale klimagassmengden som personen gjennom sine kjøp har bidratt til. Du skal ikke spesifisere *hvordan* denne metoden er implementert, men kan simpelthen anta at den klarer å beregne verdien som deretter skal vises til aktøren via kantobjektet. Du må selv vurdere hvilke metoder kantobjektet og kontrollobjektet trenger.

Her er de fleste metodene og alle objektene allerede oppgitt. Utfordringen er å kunne "tegne" dette i form av et sekvensdiagram og velge hvordan meldingene går fram og tilbake fra aktør via kantobjekt og kontrollobjekt, samt at man selvsagt må forstå ut fra oppgavebeskrivelsen av kontrollobjektet er "bm: BeregnMengde" (siden kontrollobjektet skal ha samme navn som bruksmønsteret).



Kommentar: Dette er vel omtrent den enkleste løsningen jeg kan tenke meg, hvor jeg har antatt at kontrollobjektet returnerer mengden i metoden "beregnMengdeFor(fødselsnummer)".
Man kunne også tenke seg at kontrollobjektet returnerer void eller boolean og heller sender en egen melding tilbake til kantobjektet rett etter kallet til "minMengde:=beregnMinMengde", for eks. "k.visMengde(minMengde)". Det er også en fin løsning. Her har jeg ikke med returpiler men det er selvsagt greit å ha med hvis noen ønsker det...
En annen mulighet er at kontrollobjektet ber om fødselsnummer fra kantobjektet framfor at den sendes inn som parameter. Uansett er det mange varianter her.

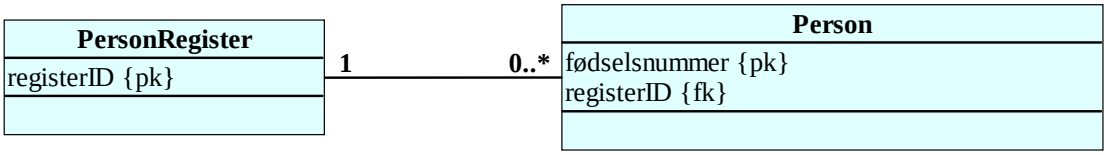
c) Tegn et klassediagram som tilsvarer sekvensdiagrammet fra oppgave 2b). Du skal KUN vise de fire klassene, metodene i hver klasse, samt assosiasjoner med multiplisitet.



Kommentar:
Her har jeg tatt med avhengighetspiler og navigering på assosiasjonen, men oppgaven ber strengt tatt ikke om det. Trekk hvis det mangler metoder i forhold til de som kandidaten har i sekvensdiagrammet

d) Lag et dataorientert klassediagram som tilsvarer klassediagrammet i oppgave 2c).

Poenget her er primært at du skal vise at du har forstått hva som faktisk skal lagres i en database (forretningsobjektene). Personregister-klassen er litt spesiell her, iom at den sannsynligvis vil inneholde kun en forekomst, så egentlig kunne den kanskje droppes, men hvis Personregister skal implementere metoden finnPerson så trenger man en slik assosiasjon hvis dette for eksempel skal realiseres via Hibernate.



Oppgave 3 (35 % – 63 min.)

Merk at de fleste svar på de følgende oppgaver kun er noen av mange muligheter. Det viktige er at du klarer å argumentere fornuftig for det du skriver og med referanser til pensum og forelesninger.

a) For å definere kravene til systemet, nedsetter Ruritaniens Direktorat for forvaltning og IKT en komité. Hvilke stakeholders mener du bør være representert i denne komiteen? Begrunn svaret ditt kort.

Forslag til svar: Foruten Direktorat for forvaltning og IKT og potensielle leverandører av systemet, bør følgende være representert:
Næringsinteresser (fordi systemet skal installeres ”overalt” hvor ting kan selges),
Forbrukere (siden systemet vil medføre en ekstra transaksjon ved nesten alle kjøp),
Datatilsynet (siden det opplagt er personvernaspekter ved innføringen av et slikt system).

b) Det er bestemt at PS2000 skal benyttes for kontraktsformål. Hvilke to stakeholders undertegner en PS2000-kontrakt, og hvem er disse i denne sammenhengen? Begrunn svaret ditt kort.

Kunde: Direktorat for forvaltning og IKT.
Leverandør: Selskapet som vinner anbudsrunden.
Det er høyst sannsynlig flere utviklingsselskaper med i starten. Det vil således være en anbudskonkurranse mellom disse selskapene.

c) Du vet at foruten de funksjonelle kravene, så er de ikke-funksjonelle kravene svært viktige å innfri dersom ditt systemforslag skal bli antatt som det endelige. Tatt i betraktning at systemet skal kunne benyttes av alle kjøpekraftige i Ruritanien på omtrent alle steder der varer eller tjenester kan kjøpes, nevn de fire ikke-funksjonelle kravene som du mener er viktigst for å skaffe deg konkurransefortrinn. Begrunn svaret ditt kort.

Løsningsforslag:
1. Installerbarhet er viktig, siden deler av systemet skal installeres ”overalt” og dette kan bare gjøres effektivt av dersom ikke-datakyndige også kan gjøre dette. Dersom installeringen oppfattes som vanskelig, vil dette være svært negativt for næringsinteressene.
2. Pålitelighet mht. korrekt registrering.
3. Sikkerhet mht. oppbevaring av konfidensielle data.

4. Brukervennlighet, siden systemet skal kunne brukes av ikke-datakyndige.

d) Hvordan vil du oppfylle disse fire kravene best mulig mht.
   - fysisk arkitektur
   - logisk arkitektur
   (Du behøver ikke å detaljere arkitekturen, men du skal si hvordan de ikke-funksjonelle kravene påvirker logisk og fysisk arkitektur.)

Forslag til svar:
Installerbarhet:
- logisk arkitektur: Installerbarhet medfører at man må sikre at systemet kan lastes opp og kjøres uten videre innstillinger av operativsystem eller hardware. Det må derfor ikke brukes komponenter (fra f.eks. andre leverandører) i systemet som krever separat installasjon, eller konsulenthjelp for å tilpasse komponentene. (Dette er omtrent samme historie som ligger bak foilene 27-30 i arkitekturforelesningen.)
– fysisk arkitektur: Systemet bør trolig bli distribuert til salgssteder som en ”hardwarekomponent” (dvs. at borgerkortleseren er ”hardkodet” med nødvendig funksjonalitet). Dette medfører en godt avgrenset klient-tjener arkitektur, noe som også vil gjenspeile seg i den logiske arkitekturen.

Pålitelighet:
- logisk arkitektur: Her bør den delen av systemet som tar seg av registrering ved varekjøp kobles mot allerede tilgjengelige databaser for persondata og varedata for å dobbeltsjekke registreringsopplysninger. (Alle slike koblinger må godkjennes av datatilsynet.) Dette medfører at systemet må integreres mot disse andre systemene. I tillegg kan man diskutere om man bør gjøre sjekken fra borgerkortleseren, eller om en forespørsel til en slik sjekk bør sendes til en sentral server, som kanskje vil være sikrere.
- fysisk arkitektur: Sammenfaller i stor grad med logisk arkitektur.

Sikkerhet:
- logisk arkitektur: Her bør kanskje databasene utformes separat av en ekstern databaseleverandør som er ekspert på sikker lagring av data.
– fysisk arkitektur: ...og databasen bør kanskje være lokalisert på en egen maskin hvis sikkerhet er ivaretatt av eksperter.
Brukervennlighet:
- logisk arkitektur: Viktig med godt GUI som er såpass løst koplet fra resten av systemet slik at forbedringer til både GUI og resten av systemet kan endres uavhengig.
– fysisk arkitektur: Utskiftning eller oppdateringer av borgerkortlesere må kunne gjøres svært enkelt.

e) Beskriv kort en overordnet utviklingsprosess for utvikling av en løsning. Gi en kort begrunnelse for forslaget ditt.

Her bør man diskutere hvor mye usikkerhet man synes det er knyttet til kravene til prosjektet, og la det styre valg av prosess. Kanskje er det slik at dette likner mye på eksisterende løsninger (som for eksempel TRUMF bonuskort) og at løsningen derfor er en enkel tilpasning av noe slikt? I så fall kan en relativt sekvensiell, spesifikasjonsdrevet fossefallsprosess være greit her. Men er det kanskje behov for integrasjon med eksisterende betalings/kassasystemer, som det finnes mange av? I så fall bør man jobbe iterativt og inkrementelt her, hvor man tar for seg en gitt

løsning først, og kanskje får den satt i prøvedrift før man ekspanderer? Se også svar under f)

f) Hva anser du som de viktigste utfordringer knyttet til endrings- og konfigurasjonsstyring i dette prosjektet?

Et slikt system vil høyst sannsynlig være i kontinuerlig evolusjon. Spesielt er det lett å se for seg at det vil bli behov for adaptive endringer når omkringliggende teknologi endrer seg. På logisk endringsnivå er det derfor viktig at det etableres en endringsprosess, som klart definerer hvordan endringsforslag samles inn, analyseres, prioriteres, gjennomføres og utrulles. Alle stakeholders må ta del i å definere denne endringsprosessen.

Sannsynligvis vil det være behov for å samtidig vedlikeholde kronologiske versjoner av serverside-tjenestene. For klientløsninger (litt uklart for meg i hvor stor grad disse inngår i løsningen) vil det også være spesielle utfordringer knyttet til bruk av ulike typer utstyr/maskinvare, slik at man muligens må konfigurasjonsstyre kombinasjonen av programvare og maskinvare.

Til sammen kan dette gi ganske store utfordringer, som bare delvis kan løses av avanserte versjons- og konfigurasjonsstyringssystemer. For å begrense kompleksiteten her er det vel så viktig å ha prinsipper for hvilke versjoner og varianter som skal vedlikeholdes over tid, og at avvikling av support gjøres på en måte som oppfattes som ryddig av brukermiljøene.

God verktøystøtte er viktig, et fornuftig endringshåndteringsverktøy bør etableres for å sikre sporbarhet fra endringskrav til implementert endring. Det må avklares om det er mulig å ha ett felles endringshåndteringsverktøy, og i så fall om det er du eller kunden, eller en tredjepart, som skal eie og drifte dette. Et versjons- og konfigurasjonssystem som gjør det enkelt å holde oversikten over varianter og versjoner, samt støtte for branching og merging på systemnivå vil være fordelaktig. Dersom det velges et avansert verktøy må man sikre seg at man har tilstrekkelig ressurser og kompetanse til å drifte verktøyet.

Til slutt - dersom man planlegger å støtte ulike varianter og versjoner underveis bør man allerede i arkitekturfasen prøve å bygge inn mekanismer som gjør dette enklere, for eksempel ved å forsøke å definere hvilke komponenter som vil være variantavhengige, og hvilke som er del av et kjernesystem.

Oppgave 4 (15 % – 27 min.)

Det er ingen tvil om at klimagass-systemet vil falle inn under Ruritaniens personopplysningslov, som er identisk med den norske.

a) Hvilke rettslige grunnlag finnes generelt for å få hjemmel til å behandle personopplysninger?

- lovhjemmel
- samtykke
- nødvendighetsgrunner

se Skagestein side 388.

b) Hvilket (eller hvilke) av disse kommer mest sannsynlig til anvendelse i forbindelse med klimagass-systemet?

Det er i dette tilfelle grunn til å tro at Ruritanien vil skaffe hjemmelen gjennom lov eller forskrift til lov.