NTNU
Kunnskap for en bedre verden

FAKULTET FOR INFORMASJONSTEKNOLOGI OG ELEKTROTEKNIKK

Institutt for Datateknologi og informatikk

Eksamensoppgave i
TDAT1005 Databaser med videregående programmering

Faglig kontakt under eksamen:
Else Lervik, tlf 482 89 200
Tore Mallaug, tlf 992 38 232

Eksamensdato: 08.06.2018

Eksamenstid (fra-til): 0900-1400

Hjelpemiddelkode/Tillatte hjelpemidler:

2 håndskrevne A4-ark som studenten selv må ta med seg til eksamen.

Annen informasjon:

Les gjennom hele oppgavesettet før du begynner arbeidet, og disponer tiden.

Dersom noe virker uklart i oppgavesettet, skal du gjøre dine egne antagelser og forklare dette i besvarelsen.

Lykke til!

Merk! Studenter finner sensur i Studentweb. Har du spørsmål om din sensur må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

OVERSIKT OVER VEDLEGG

1. Case-beskrivelsen, slik denne lå i Blackboard.
2. SQL-script for å lage databasen i Oppgave 2 (også gitt på forhånd)

Merk: For å få ståkarakter i emnet, må kandidaten ha ståkarakter på Oppgave 1, 2 og 3 hver for seg. B (eller A) forutsetter også ståkarakter på Oppgave 4.

OPPGAVE 1 – DATAMODELLERING (25%)

Dersom case-beskrivelsen dere fikk på forhånd omfatter mer, eller kan tolkes annerledes, enn det som står her, så er det det som står her som gjelder.

Dyrely Dyrehotell tilbyr følgende til sine kunder:

- Hotellopphold for kortere perioder for katter, hunder og smådyr (kaniner, fugler, hamstere, etc.)
- Barnehage (dagopphold) for hunder.
- Kurs og aktiviteter for hunder.

Du skal modellere en databaseløsning for dyrehotellet basert på følgende spesifikasjoner – sett dine egne forutsetninger dersom du mener beskrivelsen er ufullstendig:

Dyreeierne (kundene) registreres med navn, adresse, tlf og epost. De enkelte dyrene knyttes til sine eiere, en og samme eier kan ha flere dyr. Om alle dyr registreres et entydig identifikasjonsnummer, navn, fødselsår og hvorvidt dyret må bo i eget bur, eventuelt kun sammen med dyr fra samme husstand.

Eieren kan få andre til å hente dyrene sine, da skal dette være registrert i tilknytning til det enkelte dyret. Disse personene kan også være eiere til egne dyr, de registreres i databasen på samme måte som eierne (kundene).

Vi deler inn dyrene i hunder, katter og «smådyr». Med smådyr menes f.eks. kaniner og fugler.

For kattene skal det angis hvorvidt det er nødvendig å børste dem hver dag.

For alle dyr skal dato for obligatoriske vaksinasjoner registreres. Hva som er obligatoriske vaksinasjoner varierer fra dyreart til dyreart, også innen smådyrgruppen.

En dyreart deles inn i raser. Hvert enkelt dyr må koples til sin rase. En art identifiseres med et entydig nummer, mens en rase identifiseres med et entydig nummer innenfor den arten som rasen tilhører.

Dyrenes opphold på dyrehotellet skal registreres. Det enkelte oppholdet registreres med fra- og tildato.

Barnehagetilbudet er kun for hunder. Her skiller vi mellom fast plass i prosentdeler 20%, 40%, 80%, 100% og drop-in (hele dager) samt klippekort på 5, 10 eller 20 klipp. Kundenes bruk av barnehagetilbudet skal inn i databasen. En avtale registreres med et entydig nummer og datoen avtalen ble inngått. Denne datoen vil være den samme som drop-in-datoen, men den kan være en annen enn datoen hunden starter på en fast plass eller benytter det første klippet på et klippekort. For klippekortene skal alle datoer kortet ble brukt registreres.

Dyrely tilbyr også ulike kurs og aktiviteter for hunder, for eksempel valpekurs og grunnkurs. Kursene kan gå over flere dager, det er da en fast dag pr uke. Databasen skal inneholde kursbeskrivelser (navn, målgruppe, antall dager, antall timer pr dag, kort tekstlig beskrivelse, pris) og også informasjon om tidspunkter (startdato, klokkeslett) de ulike kursene kjøres.

Eierne kan melde hundene sine på kursene. De får kursbevis etter fullført kurs. Dette må inn i databasen.

Oppgave

Lag en datamodell (ER/EER) for problemstillingen. Bruk UML-notasjon.

Husk at primærnøkler alltid skal markeres i datamodellen. Fremmednøkler hører strengt tatt ikke hjemme i denne modellen, men om du ønsker kan du ta dem med. Da må du i tilfelle ta med alle, og du må markere dem med stjerne.

Oversett datamodellen til relasjonsmodellen. Sett opp relasjoner (tabeller) på relasjonell form (slik som tabellene i Oppgave 2) er satt opp). Marker primærnøkler med understrekingstegn, f.eks. _kundenr_ og fremmednøkler med stjerne.

OPPGAVE 2 –SQL (20%)

Tabellene (relasjonene) under brukes til å registrere data om hundesledeløp i ulike deler av landet på ulike år:

STED (stedid, stedsnavn)
ETAPPE (enr, fra_stedid*, til_stedid*, distanse)
LØP (lnr, aar, løpsnavn)
LØP_ETAPPE ((lnr, aar)*,enr*, løpenr)
DELTAKER (deltnr, navn, nasjonalitet, hjemsted)
TIDTAKING (((lnr, aar)*,enr*), deltnr*, tid, fullført)

Et løp består at en eller flere etapper. Hver etappe går mellom to registrerte steder og har en distanse oppgitt i antall km. Attr. løpenr er en teller (fra verdien 1) som indikerer rekkefølgen på etappene i et løp (i tilfelle etappene endres litt fra år til år vil ikke attr. enr nødvendigvis gi riktig rekkefølge i hvert løp). Løpenr = 1 vil da være startetappen i et løp, mens det største løpenummeret i et løp vil være sluttetappen. Total distanse i km for et gitt løp er summen av distansene til etappene til løpet (vi forenkler litt og sier at et løp har kun en totaldistanse; hvis det finnes flere klasser/utgaver av et løp må de registreres som separate løp i databasen).

Vi forenkler litt og sier at en deltaker alltid har samme (unike) deltakernummer (attr. deltnr) i alle løp deltakeren deltar i. Et tuppel i tabellen TIDTAKING inneholder tiden (attr. tid) deltakeren brukte på en etappe i et løp. Attr. fullført = 1 viser om en deltaker har fullført en etappe. En deltaker må fullføre alle etappene til et løp for å få løpet godkjent.

Se for øvrig vedlegg for SQL-script. I hver deloppgave under skal du skrive en eller flere SQL-spørringer (setninger) med utgangspunkt i tabellene over for å finne resultatet.

Oppgave a)

Skriv ut alle stedsnavnene løp nummer 1 var innom i 2018. Hvert stedsnavn skal skrives ut bare en gang i alfabetisk rekkefølge.

Oppgave b)

Skriv ut stedsnavnene til og fra (i stedet for stedid) og distanse for hver etappe i løp nummer 1 i 2018 i riktig rekkefølge (gitt ved attr. løpenr).

Oppgave c)

Skriv ut deltakernummer og navnet til deltakerne i løp nummer 1 i 2018 som fullførte minst 4 etapper i løpet.

Oppgave d)

Lag et view (CREATE VIEW ...) over alle deltakerne i løpet ved navn Finnmarksløpet i år 2018. View-et skal for hver deltaker inneholde deltnr samt to nye attributter; ett for antall etapper deltakeren fullførte, og ett for den totale tiden deltakerne brukte.

Oppgave e)

Skriv ut navnet på den deltakeren som vant løpet ved navn Finnmarksløpet i år 2018.

PS! Du kan bruke view-et ditt fra Oppgave 2d) i løsningen din.

OPPGAVE 3 – JAVA-PROGRAMMERING (40%)

Dersom case-beskrivelsen dere fikk på forhånd omfatter mer, eller kan tolkes annerledes, enn det som står her, så er det det som står her som gjelder.

Pass på at du ikke gjør verken mer eller mindre enn det oppgavene spør etter. Men dersom du trenger flere metoder/konstruktører for å lage det oppgavene spør etter, skal du også programmere disse. En fornuftig oppdeling i metoder ut over det oppgaven spør etter, kan gi plusspoeng ved bedømmelsen.

Husk at du for alle metoder må oppgi hvilken klasse metoden tilhører.

Du trenger ikke sette opp *import*-setninger.

I programmeringsoppgaven skal du jobbe med den delen av systemet som omfatter en kunde og hotelloppholdene for katter og hunder som denne kunden eier.

Håndtering av tilleggstjenester, f.eks. medisinering og daglige MMS av dyret, er også en del av denne oppgaven.

Du skal programmere mot database kun i siste del av denne oppgaven.

Kunde, *Dyr* og *Opphold* er sentrale klasser i denne oppgaven.

Klassen *Kunde* skal ha følgende objektvariabler:

```
class Kunde  {
  private final int kundenr;
  private final String navn;
  private ArrayList<Dyr> dyrene = new ArrayList<Dyr> dyrene();
```

Klassen *Opphold* kan enten legges som en liste i klassen *Kunde* eller som en liste i klassen *Dyr*. Klassen vil kunne ha noe ulik oppbygging avhengig av hvilken løsning du velger, men vil alltid inneholde ankomst- og avreisedato.

Oppgave a)

Denne deloppgaven består av å tegne et UML-klassediagram som dekker alle klassene i denne oppgaven. Begynn med en skisse nå, og fintegn diagrammet når du er ferdig med alle deloppgavene. Pass på at diagrammet stemmer overens med det du har programmert. Diagrammet skal dekke alle elementene som benyttes i oppgave 3b-3d.

Oppgave b)

Vi skal lage et klassetre for dyrene. Vi begrenser oss til hunder og katter. Klassetreet skal altså bestå av klassene *Dyr*, *Hund* og *Katt*.

Hvert dyr får et entydig løpenummer, anta at det eksisterer en ferdig metode for å generere dette.

I tillegg er dyrets navn entydig hos den enkelte eieren. Du kan derfor bruke navnet for å søke opp et dyr når du vet hvem eieren er.

Om en hund skal vi registrere hvorvidt den er stor, middels eller liten – bruk tallene 1, 2 og 3 for å angi størrelsen. Om kattene skal vi registrere om de må børstes hver dag eller ikke. Disse forholdene betyr noe for prisen på oppholdet:

Prisene på oppholdet er som følger:

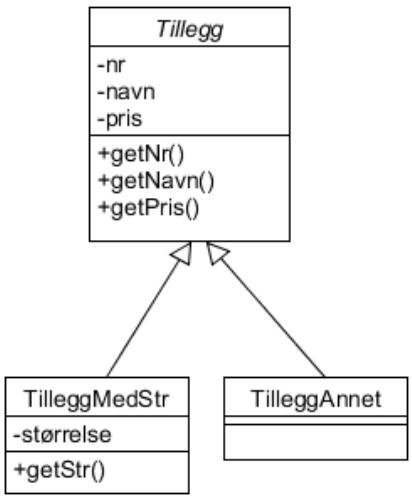
- Hunder: Kr 360 pr døgn med tillegg på hhv 30 og 50 kroner hvis hunden er størrelse middels eller stor.
- Katter: Kr 170 pr døgn med tillegg på 20 kroner hvis katten må børstes hver dag.

Programmer klassetreet med nødvendige variabler (inkl navngitte konstanter der det er hensiktsmessig), konstruktører og den polymorfe metoden *beregnNettoprisPrDøgn()*. Metoden beregner prisen for ett døgns opphold ut fra prisene gitt over.

Oppgave c)

Det er mulig å bestille tilleggstjenester for et bestemt opphold, f.eks. tannpuss, medisinering, daglige MMS etc. Prisen på de fleste tjenestene er uavhengig av dyrets størrelse, men for noen tjenester (f.eks. bading) varierer prisen med dyrets størrelse. Hunder kan som sagt være både store, middels og små. Du kan i denne sammenhengen regne at alle katter er små.

Tilleggene kan modelleres som vist på figuren nedenfor. (Tilleggstjenester er ikke del av ER/EER-modelleringen i oppgave 1.)



Et objekt hører enten til klassen *TilleggMedStr* eller klassen *TilleggAnnet*. Klassen *Tillegg* er abstrakt.

Du kan anta at disse tre klassene eksisterer. Disse klassene skal også brukes i oppgave e).

Et objekt av klassen *Opphold* inneholder informasjon om et bestemt opphold for et bestemt dyr. Ankomstdato og avreisedato skal, som sagt, med. Det skal også eksistere en liste (ArrayList) med tilleggstjenester bestilt for det enkelte oppholdet.

I en virkelig løsning vil det være naturlig å bruke dato-klasser her. Det trenger du ikke, bruk pseudokode (f.eks. fraDato.antDagerTil(tildato)).

Du skal lage klassen *Opphold* med det innhold som er nødvendig for å løse resten av oppgaven.

Lag en metode i klassen *Kunde* som registrerer et opphold for et dyr. Metoden skal returnere false dersom det nye oppholdet helt eller delvis overlapper med et eksisterende opphold for dette dyret, ellers true.

Parametere til metoden skal være en referanse til det aktuelle dyreobjektet, ankomst- og avreisedato samt en liste (ArrayList) med tillegg for dette oppholdet.

Hvis dyret ikke er registrert hos kunden (sjekk på navn), skal det registreres der.

Oppgave d)

Lag en metode i klassen *Kunde* som henter ut prisen for alle dyrene som kunden har i hotellet i en bestemt periode (fradato – tildato). Du kan anta at dyrene det gjelder er der i eksakt samme periode (samme fra- og tildato). Husk å regne med tilleggene.

Metoden skal også ta hensyn til følgende:

Dersom kunden har flere hunder og/eller flere katter inne i samme periode skal bare den første hunden/katten betale full pris. De andre får 25% rabatt. Dette skal forstås slik at man teller hunder og katter hver for seg. Én hund og én katt gir altså ikke rabatt. Dersom man derimot har inne to hunder og én katt skal man betale fullt for katten og den ene hunden. Den andre hunden får 25% rabatt. Rabatten gjelder ikke tilleggene.

Oppgave e)

Anta at vi har en klasse *Database* som tilbyr ulike tjenester mot databasen. Du skal programmere en av metodene i denne klassen:

Lag metoden *hentTillegg()* som leser dataene for tilleggstjenestene fra tabellene *tillegg* og *strTillegg* og lagrer de i riktig type objekter – se klassene *Tillegg*, *TilleggMedStr* og *TilleggAnnet* i oppgave c). Metodehodet skal se slik ut:

```
public java.util.ArrayList<Tillegg> hentTillegg()
```

Du skal bruke følgende databaseforbindelse, som du kan anta at eksisterer og er tilknyttet databasen:

```
java.sql.Connection conn;
```

Tabellene med eksempler på innhold ser slik ut:

Tabellen TILLEGG:

nr	navn	pris
1	MMS	50
2	Godteripose	25
3	Bad	300
4	Bad	400
5	Bad	500

Tabellen TILLEGG_MED_STR:

nr	størrelse
3	Liten
4	Middels
5	Stor

OPPGAVE 4 – TEORISPØRSMÅL (15%)

Oppgave a)

Gitt følgende kode:

```
public class Eksamen18v {
   private MinTest testObj;

   public  Eksamen18v(){
      testObj = new MinTest("A",1);
   }

   public MinTest getTest(){
      return testObj;
   }

   class MinTest{
      String lokalStr;
      Integer lokalNr;

      MinTest(String str, int nr){
         lokalStr = str;
         lokalNr  = nr;
      }
      public void setStr(String str){
         lokalStr = str;
      }
      public void setNr(Integer nr){
         lokalNr = nr;
      }
      public String getString(){
         return lokalStr;
      }
      public Integer getNr(){
         return lokalNr;
      }
      public String toString(){
         return lokalStr + " " + lokalNr;
      }
   }

   public static void main(String[] args) {
      Eksamen18v eksamen = new Eksamen18v();
      MinTest test = eksamen.getTest();
      System.out.println(test);

      test.setStr("B");
      test.setNr(2);
      test = eksamen.getTest();
      System.out.println(test);

      String s = test.getString();
      Integer i = test.getNr();
      s = "C";
      i = 3;
      test = eksamen.getTest();
```

```
        System.out.println(test);
    }
}
```

Hva blir resultatet når man kjører koden? Begrunn svaret.

Oppgave b)

Anta følgende databasetabell:

Lærer	**Alder**	**Emne**
Knut	29	Kjemi, Matematikk
Anne	24	Matematikk
Arne	49	Matematikk

Hvilken (om noen) normalform er denne tabellen på? Foreslå hvordan evt. tabellen kan forbedres.

Oppgave c)

Det kan være grunner for å lage egne interface. Forklar hvorfor og gi et eksempel.

VEDLEGG 1: CASE-BESKRIVELSEN, SLIK DENNE LÅ I BLACKBOARD

Inspirasjon til caset er hentet fra hjemmesiden til Skoger Dyrehotell (http://www.skogerdyrehotell.no/), men en del forenklinger er selvfølgelig gjort.

Oppgave 1) Datamodellering – 25%

Dyrely Dyrehotell tilbyr følgende til sine kunder:

- Hotellopphold for kortere perioder for katter, hunder og smådyr (kaniner, fugler, hamstere, etc.)
- Barnehage (dagopphold) for hunder.
- Kurs og aktiviteter for hunder.

Du skal modellere en databaseløsning for dyrehotellet.

Oppgave 2) SQL – 20%

Tabellene (relasjonene) under brukes til å registrere data om hundesledeløp i ulike deler av landet på ulike år:

STED (stedid, stedsnavn)
ETAPPE (enr, fra_stedid*, til_stedid*, distanse)
LØP (lnr, aar, løpsnavn)
LØP_ETAPPE ((lnr, aar)*,enr*, løpenr)
DELTAKER (deltnr, navn, nasjonalitet, hjemsted)
TIDTAKING (((lnr, aar)*,enr*), deltnr*, tid, fullført)

Et løp består at en eller flere etapper. Hver etappe går mellom to registrerte steder og har en distanse oppgitt i antall km. Attr. løpenr er en teller (fra verdien 1) som indikerer rekkefølgen på etappene i et løp (i tilfelle etappene endres litt fra år til år vil ikke attr. enr nødvendigvis gi riktig rekkefølge i hvert løp). Løpenr = 1 vil da være startetappen i et løp, mens det største løpenummeret i et løp vil være sluttetappen. Total distanse i km for et gitt løp er summen av distansene til etappene til løpet (vi forenkler litt og sier at et løp har kun en totaldistanse; hvis det finnes flere klasser/utgaver av et løp må de registreres som separate løp i databasen).

Vi forenkler litt og sier at en deltaker alltid har samme (unike) deltakernummer (attr. deltnr) i alle løp deltakeren deltar i. Et tuppel i tabellen TIDTAKING inneholder tiden (attr. tid) deltakeren brukte på en etappe i et løp. Attr. fullført = 1 viser om en deltaker har fullført en etappe. En deltaker må fullføre alle etappene til et løp for å få løpet godkjent.

Se for øvrig vedlegg for SQL-script.

Oppgave 3 Programmering – 40%

I programmeringsoppgaven skal du jobbe med den delen av systemet som omfatter en kunde og hotelloppholdene for katter og hunder som denne kunden eier. Oppgaven handler i stor grad om beregning av pris for opphold på hotellet.

Håndtering av tilleggstjenester, f.eks. medisinering og daglige MMS av dyret, er også en del av denne oppgaven.

Kunde, *Dyr* og *Opphold* er sentrale klasser i denne oppgaven.

Klassen *Kunde* skal ha følgende objektvariabler:

```
class Kunde  {
  private final int kundenr;
  private final String navn;
  private ArrayList<Dyr> dyrene = new ArrayList<Dyr> dyrene();
```

Klassen *Opphold* kan enten legges som en liste i klassen *Kunde* eller som en liste i klassen *Dyr*. Klassen vil kunne ha noe ulik oppbygging avhengig av hvilken løsning du velger, men vil alltid inneholde ankomst- og avreisedato.

VEDLEGG 2: SQL-SCRIPT (OPPGAVE 2)

```
CREATE TABLE STED (
 stedid INTEGER NOT NULL,
 stedsnavn CHAR(30) NOT NULL,
 CONSTRAINT pk_sted PRIMARY KEY(stedid)
);

CREATE TABLE ETAPPE (
 enr INTEGER NOT NULL,
 fra_stedid INTEGER NOT NULL,
 til_stedid INTEGER NOT NULL,
 distanse INTEGER NOT NULL,
 CONSTRAINT pk_etappe PRIMARY KEY(enr)
);

CREATE TABLE LOP (
 lnr INTEGER NOT NULL,
 aar YEAR NOT NULL, -- MySQLs datatype YEAR lagrer her aarstall med 4 siffer
 lopsnavn CHAR(30) NOT NULL,
 CONSTRAINT pk_lop PRIMARY KEY(lnr,aar)
);

CREATE TABLE LOP_ETAPPE (
 lnr INTEGER NOT NULL,
 aar YEAR NOT NULL,
 enr INTEGER NOT NULL,
 lopenr INTEGER,
 CONSTRAINT pk_lopeta PRIMARY KEY(lnr,aar,enr)
);

CREATE TABLE DELTAKER (
 deltnr INTEGER NOT NULL,
 navn CHAR(30) NOT NULL,
 nasjonalitet CHAR(30),
 hjemsted CHAR(30),
 CONSTRAINT pk_deltaker PRIMARY KEY(deltnr)
);


CREATE TABLE TIDTAKING (
 lnr INTEGER NOT NULL,
 aar YEAR NOT NULL,
 enr INTEGER NOT NULL,
 deltnr INTEGER NOT NULL,
 tid FLOAT, -- spesifiserer ikke datatypen for tid noe nærmere i denne oppgaven
 fullfort BOOLEAN NOT NULL DEFAULT 0,
 CONSTRAINT pk_tidtaking PRIMARY KEY(lnr,aar,enr,deltnr)
);


ALTER TABLE ETAPPE
 ADD CONSTRAINT eta_fk1 FOREIGN KEY(fra_stedid)
 REFERENCES STED (stedid);

ALTER TABLE ETAPPE
 ADD CONSTRAINT eta_fk2 FOREIGN KEY(til_stedid)
 REFERENCES STED (stedid);
```

```
ALTER TABLE LOP_ETAPPE
 ADD CONSTRAINT lopet_fk1 FOREIGN KEY(lnr,aar)
 REFERENCES LOP (lnr,aar);

ALTER TABLE LOP_ETAPPE
 ADD CONSTRAINT lopet_fk2 FOREIGN KEY(enr)
 REFERENCES ETAPPE (enr);

ALTER TABLE TIDTAKING
 ADD CONSTRAINT tidtak_fk1 FOREIGN KEY(lnr,aar,enr)
 REFERENCES LOP_ETAPPE (lnr,aar,enr);

ALTER TABLE TIDTAKING
 ADD CONSTRAINT tidtak_fk2 FOREIGN KEY(deltnr)
 REFERENCES DELTAKER (deltnr);
```