



## HØGSKOLEN I SØR-TRØNDELAG

Avdeling for informatikk og e-læring - AITeL

<b>Kandidatnr:</b>	
<b>Eksamensdato:</b>	4.mai 2011
<b>Varighet:</b>	0900-1300
<b>Emnekode:</b>	LO191D / LC191D
<b>Emnenavn:</b>	<i>Campus:</i> LC191D Videregående programmering <i>Nett:</i> LO191D Videregående programmering (i Java)
<b>Klasse(r):</b>	HING2010HA, nettstudenter, gjentak
<b>Studiepoeng:</b>	6
<b>Faglærer(e):</b>	Else Lervik og Anette Wrålsen
<b>Kontaktperson (adm.)</b>	Ingrid Island
<b>Hjelpemidler:</b>	Lærebøker, alle håndskrevne og trykte hjelpemidler.
<b>Oppgavesettet består av:</b>	4 oppgaver og 9 sider (inkludert forside og vedlegg)
<b>Vedlegg består av:</b>	3 sider
<b>Merknad: Oppgaveteksten kan beholdes av studenter som sitter eksamenstiden ut.</b>	
<b>Lykke til!</b>	

## **Les dette!**

All programmering skal skje i Java.

Det er tillatt å ha flere medlemmer i klassene enn det som er oppgitt i teksten.

Les gjennom hele oppgavesettet før du begynner å programmere. 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.

Dersom du mener det mangler opplysninger, sett dine egne forutsetninger.

Alle oppgavene arbeider med den samme problemstillingen som du får en introduksjon til nedenfor. Det vil kunne være slik at du i en oppgave med fordel kan bruke klasser/metoder du skal ha laget i en tidligere oppgave. Dersom du av en eller annen grunn ikke har laget disse klassene/metodene, kan du, når du skal løse andre oppgaver, anta at de eksisterer.

Du trenger ikke sette opp *import*-setninger.

## **Felles problemstilling for alle oppgavene**

I vedlegg 1 og 2 finner du klassene *Person* og *Sett*. De skal brukes i kode som håndterer deler av en enkel treningsdagbok. Om du ønsker kan du utvide og/eller forandre på disse klassene.

For alle treningsøkter registrerer vi øktnummer (entydig), sted, dato og varighet (minutter).

Vi skiller mellom følgende typer treningsøkter:

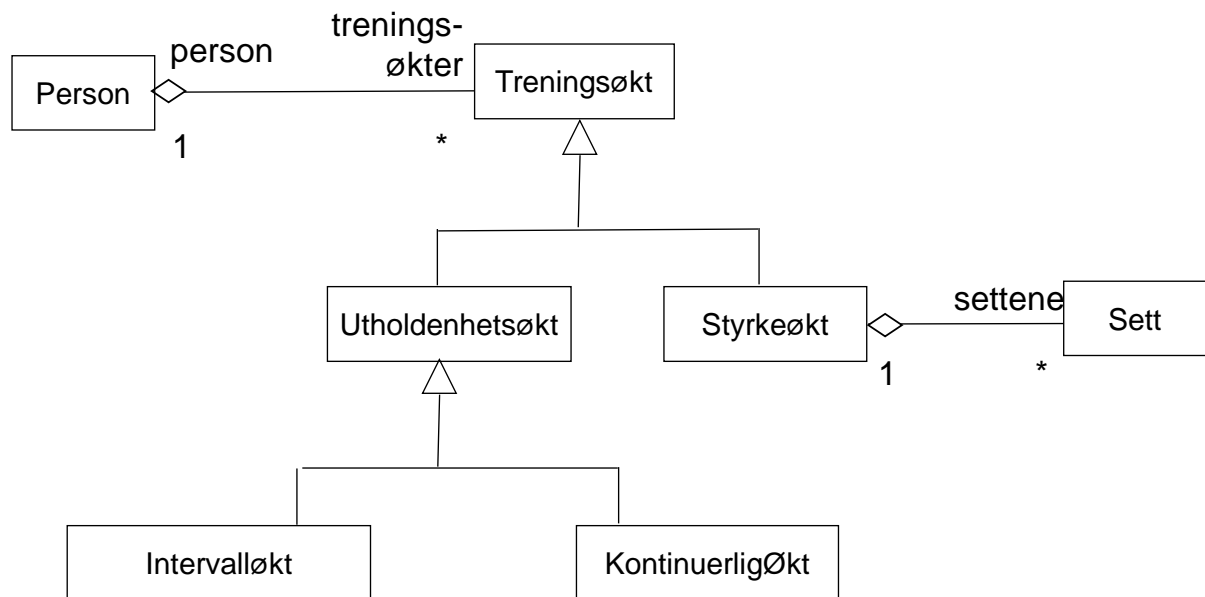
*Kontinuerlig økt*: Dette er en utholdenhetsøkt uten pauser. I tillegg til felles attributter for alle slags treningsøkter, registreres type aktivitet og tilbakelagt distanse i kilometer.

*Intervalløkt*: Dette er en utholdenhetsøkt som består av intervaller med høy intensitet og aktive pauser av fast varighet. Eksempel: 4 min høy intensitet med 3 min aktiv pause, dette utført i alt fire ganger, gir en treningstid på  $4 \times 4 = 16$  minutter. Total varighet 28 minutter. I tillegg til felles attributter for alle slags treningsøkter, registrerer vi her typen aktivitet samt intervallvarighet (i eksemplet er dette 4) og distanse tilbakelagt pr intervall (kilometer). Tilbakelagt distanse kan variere fra intervall til intervall.

*Styrkeøkt*: I tillegg til felles attributter for alle slags treningsøkter, registreres settene som økten består av. Et sett er karakterisert ved navnet på øvelsen, antall repetisjoner og antall kilo (se klassen *Sett*). Eksempel: Et sett kan bestå av øvelsen Benkpress med 20 kilo vekt og 10 repetisjoner. Det utføres aldri mer enn én styrkeøkt på en gitt dato.

Som eksempler på utholdenhetsaktiviteter ("typen aktivitet" i teksten foran) nevner vi løping, skiløping og svømming.

Du skal forholde deg til følgende sammenheng mellom klassene:



Det er tillatt å lage flere klasser om du finner det hensiktsmessig. (Men du kan ikke sløyfe noen av klassene i diagrammet.)

**OBS:**

En økt identifiseres med et nummer. Du trenger ingen steder i dette oppgavesettet sjekke og/eller generere dette nummeret slik at det virkelig er entydig. Det samme gjelder nummeret på settene.

Type aktivitet og datoer lagrer du som strenger (*String*). Heller ikke her legger du inn noen form for kontroll av gyldighet.

## Oppgave 1 – vekt 25%

Du skal programmere klassene vist i diagrammet. Husk at *Person* og *Sett* er gitt i vedlegg 1 og 2.

Les teksten foran nøye slik at du får med alle objektvariablene.

Legg merke til at sammenhengen mellom *Person* og *Treningsøkt* er toveis. Det vil si at vi i klassen *Person* har denne objektvariabelen:

```
private ArrayList<Treningsøkt> treningsøker;
```

I klassen *Treningsøkt* må du legge inn denne objektvariabelen:

```
private Person person;
```

Distansene som en intervalløkt består av, skal legges inn som en *double*-tabell.

Alle klassene skal ha en konstruktør som tar verdiene til alle aktuelle attributter som argument. (ArrayList er et ikke aktuelt attributt i denne sammenhengen.)

Alle enkle attributter skal ha get-metoder. Du kan spare skrivearbeid ved å markere dette med GET etter attributtnavnet, eksempel:

```
int antall; // ** GET
```

Men vær obs. i tilfelle noen av disse metodene redefineres i subklasser, da må du programmere dem, og bruk i tilfelle gjerne også et annet navn, for eksempel *beregnAntall()*.

Du skal ikke lage flere set-metoder enn det oppgavesettet for øvrig krever.

Du skal i denne oppgaven ikke lage metoder som opererer på ArrayListene, det kommer i de følgende oppgavene.

## Oppgave 2 – vekt 30%

Programmer metodene gitt under oppgave a) og b) nedenfor.

Ta med opplysninger om hvilke klasser metodene skal inn i (ta eventuelt også med klasser der en metode er abstrakt).

### Oppgave a:

For en hvilken som helst treningsøkt skal kaloriforbruket beregnes.

- For utholdenhetsøkter skal du multiplisere personens vekt med tilbakelagt distanse. (Dette kan synes som en merkelig formel, men den gir et godt overslag for antall forbrukte kcal.)
- For styrkeøkter antar vi at damer forbruker 240 kcal/time, mens menn forbruker 300 kcal/time.

Metoden skal ha følgende hode:

```
public double finnKaloriforbruk()
```

### Oppgave b:

Lag en metode i klassen *Person* som henter ut alle settene som tilhører styrkeøkten på en bestemt dato. (Husk kun én styrkeøkt pr. dato.)

Metoden skal ha følgende hode:

```
public Sett[] finnStyrkeøkt(String dato)
```

Metoden skal lages slik at klienten kan endre på settene. Du skal altså programmere i henhold til aggregeringsprinsippet og ikke i henhold til komposisjonsprinsippet. (Metoden skal brukes i oppgave 4.)

### Oppgave 3 – vekt 20%

Vi har gitt to databasetabeller med eksempel på datainnhold:

Tabellen *TRENINGSØKT* – bare styrkeøkter i denne oppgaven:

ØKTNR	DATO	STED	VARIGHET
1	04.05.2011	hjemme	40
2	02.05.2011	borte	35

Tabellen *SETT*:

ØKTNR	SETTNR	ØVELSE	ANTREP	KILO
1	1	knebøy	10	50
1	2	knebøy	10	50
1	3	knebøy	10	50
1	4	benkpress	10	30
.....				
2	10	knebøy	10	35
2	11	knebøy	10	40
.....				

I vedlegg 3 finner du SQL-script for å lage tabellene og fylle dem med eksempeldata.

Tabellen *Treningsøkt* inneholder altså informasjon som er aktuell for alle typer treningsøkter, mens tabellen *Sett* inneholder informasjon om de settene som styrkeøkter består av. Merk at vi her har kolonnen *ØKTNR* som identifiserer hvilken økt settet tilhører.

*Oppgave:*

I konstruktøren til klassen *Person* brukes den private metoden *lesStyrkeøkterFraDatabase()*. Du skal programmere denne metoden etter følgende spesifikasjoner:

Metodehode:

```
private void lesStyrkeøkterFraDatabase(Connection forb)
```

Databaseforbindelsen er altså gitt.

Følgende SQL-setning brukes til å hente ut alle treningsøktene (som nå altså bare er styrkeøkter):

```
SQL-setning 1: SELECT * FROM treningsøkt
```

Løp gjennom resultatsettet du får ved å kjøre SQL-setning 1. For hver resultatlinje (og hvert øktnr) slår du opp i tabellen *Sett* ved å kjøre en ny SQL-setning:

```
SQL-setning 2: SELECT * FROM sett WHERE øktNr = <øktNr>
```

Hver økt består av mange sett. Alle settene som hører til en bestemt økt, finner du altså i resultatsettet fra SQL-setning 2.

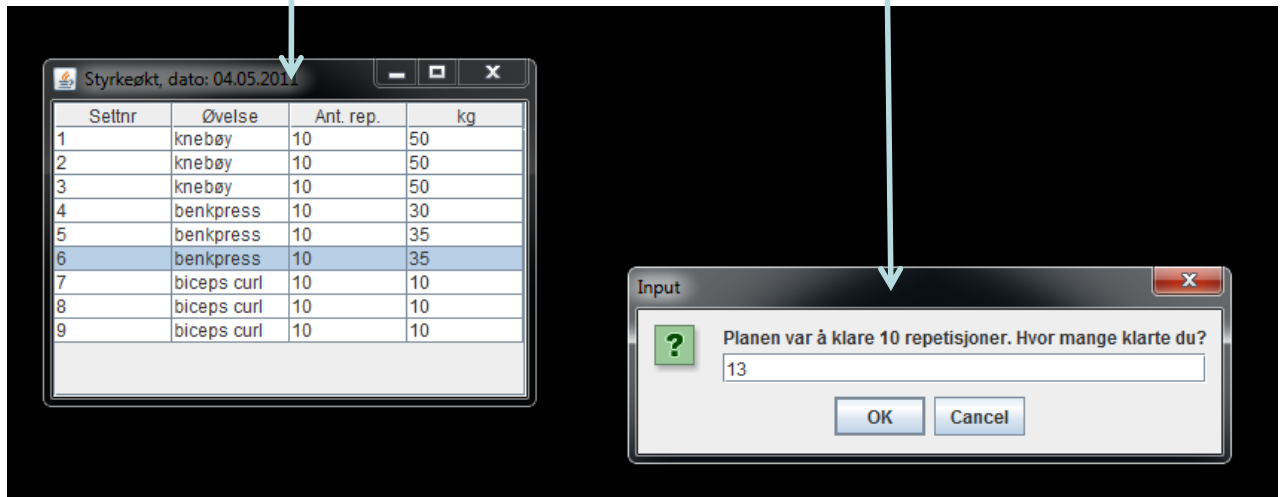
(Du må bruke et *Statement*-objekt og et *ResultSet*-objekt for hver av de to SQL-setningene.)

For hver økt skal du opprette et objekt av klassen *Styrkeøkt* og registrere alle settene som hører til. *Styrkeøkt*-objektet skal legges inn i *ArrayListen* *treningsøkter*.

## Oppgave 4 – vekt 25%

Styrkøkt på en bestemt dato  
(antar kun én styrkøkt pr. dag).  
Tittel linjen er som følger: *Styrkøkt, dato: 04.05.2010*

Linjeklikk skal gi en enkel dialog  
for å endre antall repetisjoner



Du skal programmere dette brukergrensesnittet.

Vinduet til venstre viser settene en person skal utføre i en treningsøkt en bestemt dag. Nå kan det hende personen klarer flere eller færre repetisjoner enn planlagt. Et klikk på den aktuelle linjen i tabellen til venstre skal føre til at den enkle dialogboksen til høyre vises. Her skal brukeren skrive inn riktig antall. Ved trykk på OK, skal endringene lagres (mer om det nedenfor), og også vises i tabellen til venstre.

Litt om programmeringen:

Vinduet består av kun én komponent, en *JTable*. Du trenger derfor ikke tenke på layouthåndtering.

Du skal lage koden slik at følgende testklient kan brukes:

```
public static void main(String[] args) throws Exception {
    String databasedriver = "org.apache.derby.jdbc.ClientDriver";
    Class.forName(databasedriver);
    String dbnavn = "jdbc:derby://localhost:1527/styrkedata;user=db;password=db";
    Connection forb = DriverManager.getConnection(dbnavn);
    Person pers = new Person("Anna Olsen", true, 65, forb);
    GUI gui = new GUI(pers, "04.05.2011", forb);
    gui.setVisible(true);
}
```

Databaseforbindelsen skal stenges i det vinduet lukkes.

Brukeren kan endre på antall repetisjoner i et eller flere sett på den gitte dato. Disse endringene skal vises på skjermen (tips: du trenger ikke gjøre noe spesielt for at det skal skje), og endringene skal også lagres i person-objektet. Endringene skal selvfølgelig også lagres i databasen, men her skal du **kun sette opp forslag til metodehode og plassering av metoden**. Du skal ikke programmere innholdet i metoden.

## VEDLEGG 1

```
class Person {
    private final String navn;
    private final boolean kvinne;
    private final double vekt;
    private ArrayList<Treningsokt> treningsøker = new ArrayList<Treningsokt>();

    public Person(String navn, boolean kvinne, double vekt, Connection dbForbindelse) {
        this.navn = navn;
        this.kvinne = kvinne;
        this.vekt = vekt;
        lesStyrkeøkerFraDatabase(dbForbindelse);
        // metoder som leser inn initielle data om øvrige treningsøker, ser bort fra dem her
    }

    public String getNavn() {
        return navn;
    }

    public boolean isKvinne() {
        return kvinne;
    }

    public double getVekt() {
        return vekt;
    }

    public Sett[] finnStyrkeøkt(String dato) {
        // denne skal du programmere som del av oppgave 2
    }

    private void lesStyrkeøkerFraDatabase(Connection forb) {
        // denne skal du programmere som oppgave 3
    }
}
```

## VEDLEGG 2

```
class Sett { // mulig å endre antall repetisjoner
    private final int settnr; // entydig over alle sett
    private final String øvelse;
    private int antRep;
    private final int antKilo;

    public Sett(int settnr, String øvelse, int antRep, int antKilo) {
        this.settnr = settnr;
        this.øvelse = øvelse;
        this.antRep = antRep;
        this.antKilo = antKilo;
    }

    public int getSettnr() {
        return settnr;
    }

    public String getØvelse() {
        return øvelse;
    }

    public int getAntRep() {
        return antRep;
    }

    public int getAntKilo() {
        return antKilo;
    }

    public void setAntRep(int nyAntRep) {
        antRep = nyAntRep;
    }
}
```

### VEDLEGG 3

```
CREATE TABLE treningsøkt(  
  øktNr INTEGER NOT NULL,  
  dato CHAR(10) NOT NULL UNIQUE, -- kun én styrkeøkt pr. dag  
  sted VARCHAR(20) NOT NULL,  
  varighet INTEGER NOT NULL,  
  CONSTRAINT styrke_pk PRIMARY KEY(øktNr));
```

```
CREATE TABLE sett(  
  øktNr INTEGER NOT NULL,  
  settNr INTEGER NOT NULL,  
  øvelse VARCHAR(20) NOT NULL,  
  antRep SMALLINT NOT NULL,  
  kilo SMALLINT NOT NULL,  
  CONSTRAINT sett_pk PRIMARY KEY(settNr));
```

```
ALTER TABLE sett  
  ADD CONSTRAINT sett_fk FOREIGN KEY(øktNr)  
  REFERENCES treningsøkt;
```

```
INSERT INTO treningsøkt VALUES(1, '04.05.2011', 'hjemme', 40);  
INSERT INTO treningsøkt VALUES(2, '02.05.2011', 'borte', 35);
```

```
INSERT INTO sett VALUES(1, 1, 'knebøy', 10, 50);  
INSERT INTO sett VALUES(1, 2, 'knebøy', 10, 50);  
INSERT INTO sett VALUES(1, 3, 'knebøy', 10, 50);  
INSERT INTO sett VALUES(1, 4, 'benkpress', 10, 30);  
INSERT INTO sett VALUES(1, 5, 'benkpress', 10, 35);  
INSERT INTO sett VALUES(1, 6, 'benkpress', 10, 35);  
INSERT INTO sett VALUES(1, 7, 'biceps curl', 10, 10);  
INSERT INTO sett VALUES(1, 8, 'biceps curl', 10, 10);  
INSERT INTO sett VALUES(1, 9, 'biceps curl', 10, 10);  
INSERT INTO sett VALUES(2, 10, 'knebøy', 10, 35);  
INSERT INTO sett VALUES(2, 11, 'knebøy', 10, 40);  
INSERT INTO sett VALUES(2, 12, 'knebøy', 10, 40);  
INSERT INTO sett VALUES(2, 13, 'benkpress', 10, 25);  
INSERT INTO sett VALUES(2, 14, 'benkpress', 10, 25);  
INSERT INTO sett VALUES(2, 15, 'benkpress', 10, 25);  
INSERT INTO sett VALUES(2, 16, 'biceps curl', 10, 8);  
INSERT INTO sett VALUES(2, 17, 'biceps curl', 10, 8);  
INSERT INTO sett VALUES(2, 18, 'biceps curl', 10, 8);
```