

LO191D/LC191D Videregående programmering
Løsningsforslag eksamen mai 2011

Oppgave 1

```
abstract class Treningsøkt { // klassen er abstrakt på grunn av oppgave 2a
    private final int øktNr; // identifikasjon
    private final String dato;
    private final String sted;
    private final int varighet; // antMin
    private final Person person;
    public Treningsøkt(int øktNr, String dato, String sted, int varighet, Person person) {
        this.øktNr = øktNr;
        this.dato = dato;
        this.sted = sted;
        this.varighet = varighet;
        this.person = person;
    }

    public int getØktNr() {
        return øktNr;
    }

    public String getSted() {
        return sted;
    }

    public String getDato() {
        return dato;
    }

    public int finnVarighet() { // ant min, omdefineres i Intervalløkt
        return varighet;
    }

    public Person getPerson() {
        return person;
    }
}

abstract class Utholdenhetsøkt extends Treningsøkt { // abstrakt pga oppg 2a
    private final String type; // kan med fordel være enum, men enum er ikke pensum
    public Utholdenhetsøkt(int øktNr, String dato, String sted, int varighet, Person person, String type) {
        super(øktNr, dato, sted, varighet, person);
        this.type = type;
    }
}

class KontinuerligØkt extends Utholdenhetsøkt {
    private final double distanse;
    public KontinuerligØkt(int øktNr, String dato, String sted, int varighet, Person person, String type, double distanse) {
        super(øktNr, dato, sted, varighet, person, type);
        this.distanse = distanse;
    }
}
```

```

public double getDistanse() {
    return distanse;
}
}

```

```

class Intervallokt extends Utholdenhetsokt {
    private double [] distanser;
    private int intervallVarighet;
    public Intervallokt(int øktNr, String dato, String sted, Person person, String type, double[] distanser, int intervallVarighet) {
        super(øktNr, dato, sted, distanser.length * intervallVarighet, person, type); // varighet beregnes
        this.distanser = new double[distanser.length];
        for (int i = 0; i < distanser.length; i++) {
            this.distanser[i] = distanser[i];
        }
        this.intervallVarighet = intervallVarighet;
    }
}

```

```

class StyrkeOkt extends Treningsokt {
    private ArrayList<Sett> settene = new ArrayList<Sett>();

    public StyrkeOkt(int øktNr, String dato, String sted, int varighet, Person person) {
        super(øktNr, dato, sted, varighet, person);
    }
}

```

Oppgave 2a

Klassen Treningsokt (som er abstrakt):

```

public abstract double finnKaloriforbruk();
protected boolean erKvinne() {
    return person.isKvinne();
}

protected double finnVekt() {
    return person.getVekt();
}

```

Klassen Utholdenhetsokt:

```

public double finnKaloriforbruk() {
    return finnTotalDistanse() * finnVekt();
}

```

public abstract double **finnTotalDistanse**(); // klassen blir abstrakt, metoden må programmeres i subclassene

Klassen Intervallokt:

```

public double finnTotalDistanse() {
    double sum = 0.0;
    for (int i = 0; i < distanser.length; i++) {
        sum += distanser[i];
    }
    return sum;
}

```

```
}
```

Klassen KontinuerligOkt:

```
public double finnTotalDistanse() {  
    return distanse;  
}
```

Klassen Styrkeokt:

```
private static final double KVINNE_TIMEFORBRUK = 240.0;  
private static final double MANN_TIMEFORBRUK = 300.0;  
private static final double ANT_MIN_PR_TIME = 60.0;  
public double finnKaloriforbruk() {  
    double timeforbruk = (erKvinne() ? KVINNE_TIMEFORBRUK : MANN_TIMEFORBRUK);  
    return timeforbruk * finnVarighet() / ANT_MIN_PR_TIME;  
}
```

Oppgave 2b

Klassen Person:

```
public Sett[] finnStyrkeøkt(String dato) {  
    for (Treningsøkt t : treningsøkter) { // kun én økt pr dato, stopper ved første påtreff  
        if (t instanceof StyrkeOkt && t.getDato().equals(dato)) {  
            StyrkeOkt s = (StyrkeOkt) t;  
            ArrayList<Sett> settene = s.hentSettene();  
            Sett[] settTabell = new Sett[settene.size()];  
            for (int i = 0; i < settene.size(); i++) {  
                settTabell[i] = settene.get(i);  
            }  
            return settTabell;  
        }  
    }  
    return new Sett[0];  
}
```

Klassen Styrkeokt:

```
public ArrayList<Sett> hentSettene() {  
    return settene; // klienten får muligheten til å endre settene  
}
```

Oppgave 3

```
private void lesStyrkeøkterFraDatabase(Connection forb) {  
    Statement setn1 = null;  
    Statement setn2 = null;  
    ResultSet res1 = null;  
    ResultSet res2 = null;  
    try {  
        setn1 = forb.createStatement();  
        setn2 = forb.createStatement();  
        res1 = setn1.executeQuery("SELECT * FROM treningsøkt");  
    }
```

```

while (res1.next()) {
    int øktNr = res1.getInt("øktNr");
    StyrkeOkt nyØkt = new StyrkeOkt(øktNr, res1.getString("dato"), res1.getString("sted"), res1.getInt("varighet"), this);
    res2 = setn2.executeQuery("SELECT * FROM sett WHERE øktNr = " + øktNr);
    while (res2.next()) {
        nyØkt.registrerNyttSett(
            new Sett(res2.getInt("settnr"), res2.getString("øvelse"), res2.getInt("antRep"), res2.getInt("kilo")));
    }
    treningsøker.add(nyØkt);
    Opprydder.lukkResSet(res2);
}
} catch (SQLException e) {
    Opprydder.skrivMelding(e, "Feil oppstått i lesStyrkeøkerFraDatabase()");
} finally {
    Opprydder.lukkResSet(res1);
    Opprydder.lukkSetning(setn1);
    Opprydder.lukkResSet(res2);
    Opprydder.lukkSetning(setn2);
}
}
}

```

Klassen Styrkeøkt:

```

public void registrerNyttSett(Sett sett) { // aggregeringsprinsippet, jmf. oppg. 2b
    settene.add(sett);
}

```

Oppgave 4

```

class GUI extends JFrame {
    private static final String[] kolonnenavn = {"Settnr", "Øvelse", "Ant. rep.", "kg"};
    private Connection dbForbindelse;
    private Sett[] settene;
    private Object[][] tabelldata;
    private JTable styrketabell;
    private Person person;
    private boolean dataEndret = false;
    public GUI(Person person, String dato, Connection dbForbindelse) {
        this.person = person;
        setTitle("Styrkeøkt, dato: " + dato);
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        settene = person.finnStyrkeøkt(dato); // oppgave 2b
        omformTilTabelldata();
        styrketabell = new JTable(tabelldata, kolonnenavn);

        JScrollPane rullefeltMedTabell = new JScrollPane(styrketabell);
        styrketabell.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        styrketabell.setPreferredScrollableViewportSize(new Dimension(300, 100));

        ListSelectionModel linjevalg = styrketabell.getSelectionModel();
        linjevalg.addListSelectionListener(new Linjelytter());
        addWindowListener(new Vinduslytter());

        add(rullefeltMedTabell);
    }
}

```

```

pack();
}

private void omformTilTabelldata() {
    tabelldata = new Object[settene.length][kolonnenavn.length];
    for (int settNr = 0; settNr < settene.length; settNr++) {
        tabelldata[settNr][0] = settene[settNr].getSettnr() + "";
        tabelldata[settNr][1] = settene[settNr].getØvelse();
        tabelldata[settNr][2] = settene[settNr].getAntRep() + "";
        tabelldata[settNr][3] = settene[settNr].getAntKilo() + "";
    }
}

private class Linjelytter implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent hendelse) {
        int indeks = styrketabell.getSelectedRow();
        if (indeks >= 0) {
            Sett detteSettet = settene[indeks];
            int nyAntRep = DataLeser.lesHeltall("Planen var å klare " +
                detteSettet.getAntRep() + " repetisjoner. Hvor mange klarte du?"); // DataLeser, side 496 i boka
            detteSettet.setAntRep(nyAntRep); // oppdaterer i datastrukturen
            tabelldata[indeks][2] = nyAntRep; // oppdaterer på skjermen

            // Oppdatering i databasen:
            // Dette kan gjøres på mange måter, men det er naturlig å plassere det her, slik at databasen
            // er mest mulig konsistent i forhold til datastrukturen.
            // Eksempel:
            // person.oppdaterSett(detteSettet, dbForbindelse);
            // Her utnytter vi at objektet detteSettet inneholder et entydig nr som kan brukes til å slå opp i databasen
            // Metodehode: void oppdaterSett(Sett sett, Connection dbForbindelse);
            // Alternativt lager man en metode i klassen Person som tar seg av oppdatering både i datastrukturen og i databasen.
            // Da bør klienten ikke ha direkte adgang til Sett-klassen.

            showMessageDialog(null, "Antall repetisjoner er forandret.");
            styrketabell.clearSelection();
        }
    }
}

private class Vinduslytter extends WindowAdapter {
    public void windowClosing(WindowEvent hendelse) {
        Opprydder.lukkForbindelse(dbForbindelse);
        dispose();
        System.exit(0);
    }
}
}

```