

L0191D/LC191D Videregående programmering – eksamen des. 2009

Løsningsforslag

Oppgave 1

```
abstract class Vare {
    private final int varenr;
    private final String navn;
    private double pris;
    private final int grense;
    private int kvantum;
    public static final int FAKTOR = 4;

    public Vare(int varenr, String navn, double pris, int grense, int kvantum) {
        this.varenr = varenr;
        this.navn = navn;
        this.pris = pris;
        this.grense = grense;
        this.kvantum = kvantum;
    }

    // Alle get-metoder er gitt
    public int getVarenr() {

    }

    public int finnBestillingskvantum() {
        return (kvantum < grense) ? grense * FAKTOR : 0;
    }

    public String toString() {
        return finnType() + ": " + varenr + " " + navn + ", pris: " + pris + ", kvantum på lager: " + kvantum
            + ", bestillingsmengde: " + finnBestillingskvantum() + ".";
    }

    public abstract String finnType();
}

class Torrvare extends Vare {
    public Torrvare(int varenr, String navn, double pris, int grense, int kvantum) {
        super(varenr, navn, pris, grense, kvantum);
    }

    public String finnType() {
        return "Tørrvare";
    }
}

class Ferskvare extends Vare {
    private String land;
    public static final int SPESIELLFAKTOR = 10;

    public Ferskvare(int varenr, String navn, double pris, int grense, int kvantum, String land) {
        super(varenr, navn, pris, grense, kvantum);
        this.land = land;
    }

    public String getLand() { // gitt
```

```

    return land;
}

public int finnBestillingskvantum() {
    if (getKvantum() < getGrense()) {
        return (spesiellPeriode()) ? getGrense() * SPESIELLFAKTOR : getGrense() * FAKTOR;
    } else {
        return 0;
    }
}

public String finnType() {
    return "Ferskvare";
}

private boolean spesiellPeriode() { // gitt
    ...
}
}

```

Oppgave 2

```

/**
 * Konstruktøren leser inn data fra databasen, oppretter vareobjekter og legger disse inn i ArrayListen.
 */
public Butikk() {
    Connection forbindelse = åpneForbindelse();
    Statement setning = null;
    ResultSet res = null;
    try {
        setning = forbindelse.createStatement();

        /*
         * Alle tørrvarene.
         */
        res = setning.executeQuery(
            "SELECT * FROM vare WHERE varenr NOT IN (SELECT varenr FROM ferskvare) ORDER BY varenr");
        while (res.next()) {
            Torrvare nyVare = new Torrvare(res.getInt("varenr"), res.getString("navn"),
                res.getDouble("pris"), res.getInt("grense"), res.getInt("kvantum"));
            varer.add(nyVare);
        }
        Opprydder.lukkResSet(res);

        /*
         * Alle ferskvarene
         */
        res = setning.executeQuery(
            "SELECT vare.*, land FROM vare, ferskvare WHERE vare.varenr = ferskvare.varenr "
            + "ORDER BY vare.varenr");
        while (res.next()) {
            Ferskvare nyVare = new Ferskvare(res.getInt("varenr"), res.getString("navn"), res.getDouble("pris"),
                res.getInt("grense"), res.getInt("kvantum"), res.getString("land"));
            varer.add(nyVare);
        }
    } catch (SQLException e) {
        Opprydder.skrivMelding(e, "Konstruktør Butikk()");
    } finally {
        Opprydder.lukkResSet(res);
        Opprydder.lukkSetning(setning);
        Opprydder.lukkForbindelse(forbindelse);
    }
}

```

Oppgave 3

```
public double finnVerdiVarebeholdning() {
    double sum = 0;
    for (Vare v: varer) {
        sum += v.getKvantum() * v.getPris();
    }
    return sum;
}

public String lagBestillingsliste() {
    String liste = "";
    for (Vare v : varer) {
        int best = v.finnBestillingskvantum();
        if (best > 0) {
            liste += v.getVarenr() + " " + v.getNavn() + ", bestill: " + best + "\n";
        }
    }
    return liste;
}
```

Oppgave 4

Lager en ny klasse:

```
class LandInfo implements Comparable<LandInfo> {
    private String land;
    private int antall = 0;

    public LandInfo(String land) {
        this.land = land;
    }

    public void økAntall() {
        antall++;
    }

    public String getLand() {
        return land;
    }

    public int getAntall() {
        return antall;
    }

    public String toString() {
        return antall + " " + land;
    }

    /* Til bruk ved sortering etter stigende antall */
    public int compareTo(LandInfo detAndre) {
        return (detAndre.antall - antall);
    }
}
```

Her kommer metoden i klassen Butikk:

```
public ArrayList<LandInfo> finnLand() {
```

```

ArrayList<LandInfo> landliste = new ArrayList<LandInfo>();
for (Vare v : varer) {
    if (v instanceof Ferskvare) {
        Ferskvare fv = (Ferskvare) v;
        int indeks = finnLand(landliste, fv.getLand()); // se nedenfor
        if (indeks >= 0) {
            landliste.get(indeks).økAntall();
        } else {
            LandInfo nyttLand = new LandInfo(fv.getLand());
            nyttLand.økAntall(); // setter antall lik 1
            landliste.add(nyttLand);
        }
    }
}
Collections.sort(landliste);
return landliste;
}

```

```

// Hører egentlig ikke til klassen Butikk. Kunne ev. vært static.
private int finnLand(ArrayList<LandInfo> liste, String land) {
    for (int i = 0; i < liste.size(); i++) {
        if (liste.get(i).getLand().equals(land)) {
            return i;
        }
    }
    return -1; // land ikke registrert
}

```

Oppgave 5

```

class GUI extends JFrame {
    private Butikk butikken;
    private DefaultListModel listeinnhold = new DefaultListModel();
    private JList vareliste = new JList(listeinnhold);

    public GUI(Butikk butikken) {
        super("Vareliste");
        for (Vare v : butikken.getVarer()) { // ekstra metode i klassen Butikk, se nedenfor
            listeinnhold.addElement(v);
        }
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new JLabel("Velg en vare:"), BorderLayout.NORTH);
        JScrollPane rullefeltMedListe = new JScrollPane(vareliste);
        add(rullefeltMedListe, BorderLayout.CENTER);
        vareliste.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        vareliste.addListSelectionListener(new ListeLytter());
        pack();
    }

    /* Lytteren fanger opp alle klikk på linjer i tabellen. */
    private class ListeLytter implements ListSelectionListener {
        public void valueChanged(ListSelectionEvent hendelse) {
            Vare v = (Vare) vareliste.getSelectedValue();
            if (v != null) {
                String svar = showInputDialog("Oppgi endring i kvantum: ");
                int endring = 0;
                if (svar != null) {
                    try {
                        endring = Integer.parseInt(svar);
                    } catch (NumberFormatException e) { // ignorerer ev. feilinput
                    }
                }
            }
        }
    }
}

```

```

    }
    if (v.endreKvantum(endring)) { // ekstra metode i klassen Vare, se nedenfor
        showMessageDialog(null, "Endring utført, nytt kvantum: " + v.getKvantum());
    } else {
        showMessageDialog(null, "Endringen vil gi negativ lagerbeholdning. Utføres derfor ikke.");
    }
    vareliste.clearSelection();
}
}
}
}
}

```

Det er mulig å løse oppgaven uten å bruke DefaultListModel:

```

private JList vareliste;
Inni konstruktøren:
ArrayList<Vare> varene = butikken.getVarer();
Object [] objekter = varene.toArray(); -- kan ev. bruke løkke
vareliste = new JList(objekter);

```

Ny metode i klassen Butikk:

```

ArrayList<Vare> getVarer() { // kun pakkeadgang for å beskytte dataene, brukes i GUI
    return varer;
}

```

Ny metode i klassen Vare:

```

/**
 * Hvis endringen medfører at kvantum blir negativt, utføres endringen ikke.
 * Da returnerer metoden false. Ellers utføres endringen, og metoden returnerer true.
 */
public boolean endreKvantum(int endring) {
    if (kvantum + endring >= 0) {
        kvantum += endring;
        return true;
    } else {
        return false;
    }
}
}

```