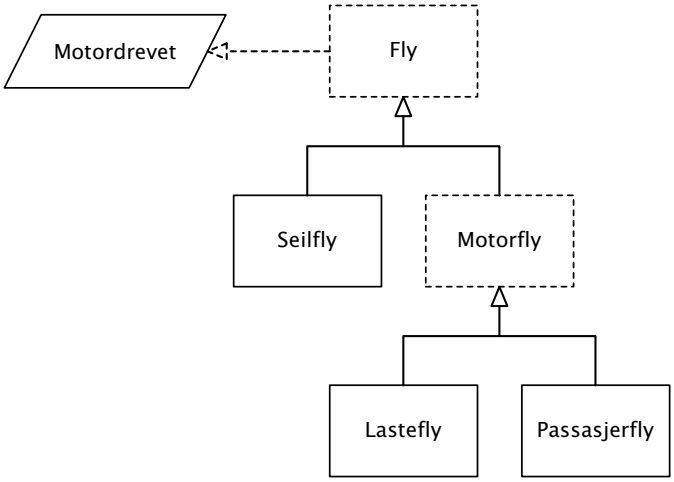
Eksamen i IN1010

Eric Jul* Dag Langmyhr†

5. juni 2024 kl 9:00–13:00

Dette er et løsnings*forslag* så det finnes andre løsninger som er like gode og som vil gi full uttelling.

Oppgave 1



Oppgave 2–5

```
import java.util.Iterator;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

// Oppgave 2

interface Motordrevet {
    int trekkraft ();
}

abstract class Fly implements Motordrevet {
    final String id;
    final int motorAntall, MTOW, kraft;
    Fly neste = null;

    Fly (String ident, int antall, int vekt, int kW) {
        id = ident;  motorAntall = antall;  MTOW = vekt;  kraft = kW;
    }

    String hentId () { return id; }
    int hentMotorAntall () { return motorAntall; }
```

*ericbj@ifi.uio.no
†dag@ifi.uio.no

```
    int hentMTOW () { return MTOW; }

    @Override
    public int trekkraft () {
        return kraft;
    }

    // Oppgave 4b

    int finnMaksVektR () {
        if (neste == null)
            return MTOW;

        int maxResten = neste.finnMaksVektR();
        if (MTOW > maxResten)
            return MTOW;
        else
            return maxResten;
    }
}

abstract class MotorFly extends Fly {
    MotorFly (String ident, int antall, int vekt, int kW) {
        super(ident, antall, vekt, kW);
    }
}

class LasteFly extends MotorFly {
    final int lastevekt;

    LasteFly (String ident, int antall, int vekt, int kW, int last) {
        super(ident, antall, vekt, kW);
        lastevekt = last;
    }
}

class PassasjerFly extends MotorFly {
    final int maksPassasjerer;

    PassasjerFly (String ident, int antall, int vekt, int kW, int maksPass) {
        super(ident, antall, vekt, kW);
        maksPassasjerer = maksPass;
    }
}

class SeilFly extends Fly {
    final int minSynkehastighet;

    SeilFly (String ident, int antall, int vekt, int kW, int synk) {
        super(ident, antall, vekt, kW);
        minSynkehastighet = synk;
    }
}

// Oppgave 3 og 4

class Flyformasjon implements Iterable<Fly> {
    // Oppgave 3a

    Fly førsteFly = null;
```

```
Flyformasjon () { /* tom */ }

// Oppgave 3b

void leggTil (Fly nytt) {
    nytt.neste = førsteFly;
    førsteFly = nytt;
}

// Oppgave 3c

boolean erMed (String sjekkId) {
    Fly f = førsteFly;
    while (f != null) {
        if (f.hentId().equals(sjekkId)) return true;
        f = f.neste;
    }
    return false;
}

// Oppgave 3d

Fly taUt (String fjernId) {
    Fly detteFly = førsteFly, forrigeFly = null;
    while (detteFly != null) {
        if (detteFly.hentId().equals(fjernId)) {
            if (forrigeFly == null) {
                // Fjern første fly i formasjonen:
                førsteFly = førsteFly.neste;
                return detteFly;
            } else {
                // Fjern et fly som ikke er først:
                forrigeFly.neste = detteFly.neste;
                return detteFly;
            }
        }
        forrigeFly = detteFly;  detteFly = detteFly.neste;
    }
    return null;
}

// Oppgave 3e

@Override
public FlyIterator iterator () {
    return new FlyIterator();
}

class FlyIterator implements Iterator<Fly> {
    Fly detteFly;

    FlyIterator () {
        detteFly = førsteFly;
    }

    @Override
    public boolean hasNext () {
        return detteFly != null;
    }
```

```
        @Override
        public Fly next () {
            Fly res = detteFly;
            detteFly = detteFly.neste;
            return res;
        }
    }

    // Oppgave 3f

    PassasjerFly[] hentPassasjerFly () {
        // Tell først antall passasjerfly:

        int antPassasjerFly = 0;
        for (Fly f: this) {
            if (f instanceof PassasjerFly) antPassasjerFly++;
        }

        // Lag så arrayen og fyll den opp:
        PassasjerFly[] res = new PassasjerFly[antPassasjerFly];
        int n = 0;
        for (Fly f: this) {
            if (f instanceof PassasjerFly) {
                res[n] = (PassasjerFly)f;  n++;
            }
        }
        return res;
    }

    // Oppgave 4a

    int totalVekt () {
        int res = 0;
        for (Fly f: this) {
            res += f.hentMTOW();
        }
        return res;
    }

    // Oppgave 4b

    int maksVekt () {
        if (førsteFly == null)
            return 0;
        return førsteFly.finnMaksVektR();
    }
}

// Oppgave 5a

class Rullebane {
    int antVentendeFly = 0;
    Lock laas = new ReentrantLock();
    Condition avventStartTillatelse = laas.newCondition();

    void sjekkAvganger () {
        laas.lock();
        try {
            if (antVentendeFly == 0) return;
```

```
            avventStartTillatelse.signal();
            antVentendeFly--;
        } finally {
            laas.unlock();
        }
    }

    void hentStartTillatelse (Fly f) {
        laas.lock();
        try {
            antVentendeFly++;
            avventStartTillatelse.await();
        } catch (InterruptedException e) {
            return;
        } finally {
            laas.unlock();
        }
    }

    // Oppgave 5d

    Condition avventFly = laas.newCondition();

    void sjekkAvganger5d () {
        laas.lock();
        try {
            while (antVentendeFly == 0) avventFly.await();
            avventStartTillatelse.signal();
            antVentendeFly--;
        } catch (InterruptedException e) {
            return;
        } finally {
            laas.unlock();
        }
    }

    void hentStartTillatelse5d (Fly f) {
        laas.lock();
        try {
            antVentendeFly++;
            avventFly.signalAll();
            avventStartTillatelse.await();
        } catch (InterruptedException e) {
            return;
        } finally {
            laas.unlock();
        }
    }
}

// Oppgave 5b

class Flygeleder implements Runnable {
    Rullebane minRullebane;

    Flygeleder (Rullebane r) {
        minRullebane = r;
    }

    @Override
```

```
    public void run () {
        while (true) {
            minRullebane.sjekkAvganger();
            try {
                Thread.sleep(60 * 1000);  /* = 1 minutt */
            } catch (InterruptedException e) { return; }
        }
    }
}

// Oppgave 5c

class Pilot implements Runnable {
    Fly mittFly;
    Rullebane minRullebane;

    Pilot (Rullebane r, Fly f) {
        minRullebane = r;  mittFly = f;
    }

    @Override
    public void run () {
        minRullebane.hentStartTillatelse(mittFly);
    }
}
```