## Løsningsforslag - Tog

A diagram of a company

Description automatically generated

**Oppgave 2-5**

// Oppgave 2–5

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import java.util.ArrayList;

import java.util.Iterator;

// Oppgave 2

abstract class Skinnegående {

    private final int sporvidde;

    private final String id;

    Skinnegående neste = null, forrige = null; // til dobbeltlenket liste

    Skinnegående (int spv, String ident) {

        sporvidde = spv;

        id = ident;

    }

    int hentSporvidde () {

        return sporvidde;

    }

    String hentId () {

        return id;

    }

    // Oppgave 4d

    void sjekkDenneSporviddenR () {

        if (neste == null) return;

        if (sporvidde != neste.sporvidde)

            throw new FeilSporvidde();

        neste.sjekkDenneSporviddenR();

    }

}

interface Motordrevet {

    boolean fossilt ();

    int trekkraft ();

}

class Lokomotiv extends Skinnegående implements Motordrevet {

    private final boolean erFossilt;

    private final int kraft;

    Lokomotiv (int spv, String ident, boolean fos, int tkraft) {

        super(spv, ident);

        erFossilt = fos;

        kraft = tkraft;

    }

    @Override

    public boolean fossilt () { return erFossilt; }

    @Override

    public int trekkraft () { return kraft; }

    }

abstract class Vogn extends Skinnegående {

    final int lengde;

    Vogn (int spv, String ident, int len) {

        super(spv, ident);

        lengde = len;

    }

}

class Godsvogn extends Vogn {

    final double lastevekt;

    Godsvogn (int spv, String ident, int len, double vekt) {

        super(spv, ident, len);

        lastevekt = vekt;

    }

}

class Passasjervogn extends Vogn {

    final int maxPassasjerer;

    Passasjervogn (int spv, String ident, int len, int maxPass) {

        super(spv, ident, len);

        maxPassasjerer = maxPass;

    }

}

// Oppgave 3

class Tog implements Iterable<Skinnegående> {

    // Oppgave 3a

    Skinnegående første = null, siste = null;

    Tog () {

    // Trengs egentlig ikke.

    }

    // Oppgave 3b

    void leggTil (Skinnegående s) {

        if (første == null) {

            første = siste = s;

        } else {

            siste.neste = s;

            s.forrige = siste;

            siste = s;

        }

    }

    // Oppgave 3c

    Skinnegående taUt (Skinnegående s) {

        if (s == første && s == siste) {

            // Kun ett element i listen.

            første = siste = null;

        } else if (s == første) {

            første = første.neste;

            første.forrige = null;

        } else if (s == siste) {

            siste = siste.forrige;

            siste.neste = null;

        } else {

            s.forrige.neste = s.neste;

            s.neste.forrige = s.forrige;

        }

        s.forrige = s.neste = null;

        return s;

    }

    // Oppgave 3d

    private Skinnegående finn (String id) {

        // Ikke bedt om, men nyttig.

        Skinnegående sx = første;

        while (sx != null) {

            if (sx.hentId().equals(id))

                return sx;

            sx = sx.neste;

        }

        return null;

    }

    Skinnegående finnOgTaUt (String id) {

        Skinnegående s = finn(id);

        if (s == null) return null;

        return taUt(s);

    }

    // Oppgave 3e

    void leggTilForan (Skinnegående s1, Skinnegående s2) {

        s2.neste = s1; s2.forrige = s1.forrige;

        if (s1 == første) {

            første = s2;

        } else {

            s1.forrige.neste = s2;

        }

        s1.forrige = s2;

    }

    // Oppgave 3f

    public Iterator<Skinnegående> iterator () {

        return new TogIterator();

    }

    class TogIterator implements Iterator<Skinnegående> {

        Skinnegående sIter = første;

        @Override

        public boolean hasNext () {

            return sIter != null;

        }

        @Override

        public Skinnegående next () {

            Skinnegående res = sIter;

            sIter = sIter.neste;

            return res;

        }

    }

    // Oppgave 3g

    Passasjervogn[] hentPassasjervogner () {

        int antPvogner = 0;

        for (Skinnegående s: this)

            if (s instanceof Passasjervogn) ++antPvogner;

        Passasjervogn[] res = new Passasjervogn[antPvogner];

        int resAnt = 0;

        for (Skinnegående s: this)

            if (s instanceof Passasjervogn)

                res[resAnt++] = (Passasjervogn)s;

        return res;

    }

    // Oppgave 4b

    void sjekkSporvidde () {

        if (første == null) return;

        int vidde1 = første.hentSporvidde();

        for (Skinnegående sx: this)

            if (sx.hentSporvidde() != vidde1)

                throw new FeilSporvidde();

    }

    // Oppgave 4c

    void leggTilSikker (Skinnegående s) {

        if (første != null) {

            sjekkSporvidde();

            if (første.hentSporvidde() != s.hentSporvidde())

                throw new FeilSporvidde();

        }

        leggTil(s);

    }

    // Oppgave 4d

    void sjekkSporviddeR () {

        if (første != null)

            første.sjekkDenneSporviddenR();

    }

}

// Oppgave 4a

class FeilSporvidde extends RuntimeException {}

// Oppgave 5a

class Leter implements Runnable {

    Tog tog;

    Monitor monitor;

    String tekst;

    Leter (Tog t, Monitor mon, String s) {

        tog = t; monitor = mon; tekst = s;

    }

    @Override

    public void run () {

        for (Skinnegående sx: tog) {

            if (sx.hentId().startsWith(tekst))

                monitor.leggTil(sx);

        }

        monitor.ferdigLeting();

    }

}

// Oppgave 5b

class Resultat implements Runnable {

    Monitor monitor;

    Resultat (Monitor mon) {

        monitor = mon;

    }

    @Override

    public void run () {

        try {

            while (true) {

                Skinnegående s = monitor.hentNeste();

                if (s == null) break;

                System.out.println(s);

            }

        } catch (InterruptedException e) {

            System.exit(1);

        }

    }

}

// Oppgave 5c

class Monitor {

    private int antLetereIgjen;

    private ArrayList<Skinnegående> buffer = new ArrayList<>();

    private Lock laas = new ReentrantLock();

    private Condition noeSkjer = laas.newCondition();

    Monitor (int ant) {

        antLetereIgjen = ant;

    }

    void leggTil (Skinnegående s) {

        laas.lock();

        try {

            buffer.add(s);

            noeSkjer.signal();

        } finally {

            laas.unlock();

        }

    }

    void ferdigLeting () {

        laas.lock();

        try {

            --antLetereIgjen;

            noeSkjer.signal();

        } finally {

            laas.unlock();

        }

    }

    Skinnegående hentNeste () throws InterruptedException {

        laas.lock();

        try {

            while (buffer.size()==0 && antLetereIgjen>0) {

                noeSkjer.await();

            }

            if (buffer.size() > 0)

                return buffer.remove(0);

            return null;

        } finally {

            laas.unlock();

        }

    }

}