OBLIG 3 – LISTER

Lenkeliste.java

abstract public class Lenkeliste<T> implements Liste<T> {

    public class Node {

        private T verdi;

        private Node neste;

        public Node(T verdi) {

            this.verdi = verdi;

        }

        public Node hentNeste() {

            return neste;

        }

        public void settNeste(Node n) {

            neste = n;

        }

        public T hentVerdi() {

            return verdi;

        }

    }

    protected int stoerrelse = 0;

    protected Node forste;

    protected Node siste;

    public int stoerrelse() {

        return stoerrelse;

    }

    // legger til nytt node objekt bakerst i listen

    public void leggTil(T x) {

        Node nyNode = new Node(x);

        stoerrelse++;

        if (stoerrelse == 1) { // listen var tom

            forste = nyNode;

            siste = nyNode;

            return;

        }

        siste.settNeste(nyNode);

        siste = nyNode;

    }

    // fjerner forste element i lenket liste og returnerer det

    public T fjern() throws UgyldigListeindeks {

        if (forste == null) { // hvis listen er tom...

            throw new UgyldigListeindeks(0);

        }

        Node kopi = forste;

        forste = forste.hentNeste(); // setter forstes verdi til aa peke paa det neste elementet

        stoerrelse--;

        return kopi.hentVerdi();

    }

}

IndeksertListe.java

public class IndeksertListe<T> extends Lenkeliste<T> {

    public void leggTil(int pos, T x) throws UgyldigListeindeks {

        if (pos > stoerrelse || pos < 0) {

            throw new UgyldigListeindeks(pos);

        }

        Node nyNode = new Node(x); // nyNode skal bli pekt paa av pos-1 og skal peke paa pos+1

        Node kopi = forste;

        stoerrelse++;

        if (pos == 0) { // vi skal sette noden forrerst i listen

            if (stoerrelse == 1) {siste = nyNode;} // listen er tom

            forste = nyNode;

            nyNode.settNeste(kopi);

            return;

        }

        for (int i = 0; i < pos - 1; i++) {

            kopi = kopi.hentNeste();

        }

        // kopi er node paa pos

        nyNode.settNeste(kopi.hentNeste()); // nyNode peker paa node paa pos+1

        kopi.settNeste(nyNode); // node paa pos-1 peker naa paa nyNode

        if (pos == stoerrelse-1) {

            siste = nyNode;

        }

    }

    public void sett(int pos, T x) throws UgyldigListeindeks {

        if (pos >= stoerrelse || pos < 0) {

            throw new UgyldigListeindeks(pos);

        }

        Node nyNode = new Node(x);

        Node kopi = forste;

        for (int i = 0; i < pos - 1; i++) {

            kopi = kopi.hentNeste();

        }

        nyNode.settNeste(kopi.hentNeste().hentNeste());

        kopi.settNeste(nyNode);

        if (pos == stoerrelse) {

            siste = nyNode;

        }

    }

    public T hent(int pos) throws UgyldigListeindeks {

        if (pos > stoerrelse || pos < 0) {

            throw new UgyldigListeindeks(pos);

        }

        Node kopi = forste;

        // her vil vi bare ha det paa pos og er ikke interessert i pos-1, pos+1

        // dermed i < pos i stedet for i < pos - 1

        for (int i = 0; i < pos; i++) {

            kopi = kopi.hentNeste();

        }

        return kopi.hentVerdi();

    }

    public T fjern(int pos) throws UgyldigListeindeks {

        if (pos >= stoerrelse || pos < 0) {

            throw new UgyldigListeindeks(pos);

        }

        Node kopi = forste;

        for (int i = 0; i < pos - 1; i++) {

            kopi = kopi.hentNeste();

        }

        T kopisVerdi = kopi.hentNeste().hentVerdi();

        kopi.settNeste(kopi.hentNeste().hentNeste());

        stoerrelse--;

        return kopisVerdi;

    }

}

Prioritetskoe.java

public class Prioritetskoe<T extends Comparable<T>> extends IndeksertListe<T> {

    @Override

    public void leggTil(T x) {

        if (stoerrelse == 0) {

            super.leggTil(x); // Lenkeliste sin leggTil

            return;

        }

        Node kopi = forste;

        for (int i = 0; i < stoerrelse; i++) {

            if (i != 0) {

                kopi = kopi.hentNeste();

            }

            if (kopi.hentVerdi().compareTo(x) > 0) {

                super.leggTil(i, x); // IndeksertListe sin leggTil

                return;

            }

        }

        super.leggTil(x);

    }

}

OBLIG 5 – TRÅDER

Oblig5Hele.java

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

import java.util.ArrayList;

import java.util.HashMap;

public class Oblig5Hele {

    // hvor mange traader som jobber med fletting, og lesing (antTraader\*2 vil

    // vaere hvor mange traader som kjoeres samtidig i dette programmet)

    private static final int antTraader = 8;

    public static void main(String[] args) {

        Monitor2 syk = new Monitor2();

        Monitor2 frisk = new Monitor2();

        Monitor2[] monitorer = {syk, frisk};

        try {

            lesFraMappe(args[0], monitorer);

            Thread traader[] = new Thread[antTraader\*2];

            for (int i = 0; i < antTraader; i++) {

                traader[i] = new Thread(new FletteTrad(syk));

                traader[i].start();

            }

            for (int i = 8; i < antTraader\*2; i++) {

                traader[i] = new Thread(new FletteTrad(frisk));

                traader[i].start();

            }

            // venter paa at alle traadene skal bli ferdig

            for (Thread traad : traader) {

                traad.join();

            }

            // hvis filene i mappen ikke har sannhetsverdier

            if (syk.hentStoerrelse() == 0) {

                int hoyestAnt = 0;

                String hoyestStreng = "";

                HashMap<String, Subsekvens> map = frisk.hentFraIndex(0);

                for (HashMap.Entry<String, Subsekvens> entry : map.entrySet()) {

                    if (entry.getValue().hentAntall() > hoyestAnt) {

                        hoyestAnt = entry.getValue().hentAntall();

                        hoyestStreng = entry.getKey() + " har flest forekomster med " + hoyestAnt;

                    }

                }

                System.out.println(hoyestStreng);

                System.exit(0); // avslutt

            }

            HashMap<String, Subsekvens> sykMap = syk.hentFraIndex(0); // invariant: stoerrelse er naa = 1

            HashMap<String, Subsekvens> friskMap = frisk.hentFraIndex(0);

            for (HashMap.Entry<String, Subsekvens> entry : sykMap.entrySet()) {

                try {

                    if (entry.getValue().hentAntall() >= friskMap.get(entry.getKey()).hentAntall() + 7) {

                        System.out.println(String.format("%s har vesentlig fler forekomster (%d) fra syk til frisk",

                            entry.getKey(),

                            entry.getValue().hentAntall() - friskMap.get(entry.getKey()).hentAntall()

                            )

                        );

                    }

                } catch (NullPointerException e) {

                    continue;

                }

            }

            System.exit(0); // avslutt program naar main traad har kommet hit

        } catch (IndexOutOfBoundsException e) {

            System.out.println("Vennligst oppgi en mappe");

            e.printStackTrace();

        } catch (InterruptedException e) {

            System.out.println("Traad avbrutt");

            e.printStackTrace();

        }

    }

    public static void lesFraMappe(String mappeNavn, Monitor2[] monitorer) {

        try {

            File fil = new File(mappeNavn + "/metadata.csv");

            Scanner leser = new Scanner(fil);

            ArrayList<Thread> traader = new ArrayList<>();

            while (leser.hasNextLine()) {

                String linje = leser.nextLine();

                if (linje.contains("amino\_acid")) {continue;} // ikke lag en traad for forste linje av filer i data mappen

                Runnable leseTraad;

                if (linje.substring(linje.length() - 4).equals("True")) {

                    leseTraad = new LeseTrad(mappeNavn + "/" + linje.split(",")[0], monitorer[0]);

                } else if (linje.substring(linje.length() - 5).equals("False")) {

                    leseTraad = new LeseTrad(mappeNavn + "/" + linje.split(",")[0], monitorer[1]);

                } else {

                    leseTraad = new LeseTrad(mappeNavn + "/" + linje, monitorer[1]);

                }

                Thread traad = new Thread(leseTraad);

                traad.start();

                traader.add(traad);

            }

            for (Thread traad : traader) {

                traad.join();

            }

            leser.close();

        } catch (FileNotFoundException e) {

            System.out.println("Fant ikke filen.");

            e.printStackTrace();

        } catch (InterruptedException e) {

            System.out.println("Traad avbrutt");

            e.printStackTrace();

        }

    }

}

Monitor2.java

import java.util.concurrent.locks.\*;

import java.util.HashMap;

import java.util.ArrayList;

public class Monitor2 extends SubsekvensBeholder {

    private Lock laas = new ReentrantLock(true);

    private Condition ikkeTom = laas.newCondition();

    public void flett() {

        laas.lock();

        try {

            if (hentStoerrelse() > 1) {

                HashMap<String, Subsekvens> map1 = hentFraIndex(0);

                HashMap<String, Subsekvens> map2 = hentFraIndex(1);

                fjernMap(map1); // fjerner saa samme maps ikke flettes to ganger

                fjernMap(map2);

                HashMap<String, Subsekvens> flettetMap = flettToMaps(map1, map2);

                hentListe().add(flettetMap);

            }

        } finally {

            laas.unlock();

        }

    }

    @Override

    public void settInn(HashMap<String, Subsekvens> map) {

        laas.lock();

        try {

            super.settInn(map);

            if (hentStoerrelse() > 1){

                ikkeTom.signal();

            }

        } finally {

            laas.unlock();

        }

    }

}

Oblig6 – Rekursjon  
Labyrint.java

import java.io.File;

import java.io.FileNotFoundException;

import java.lang.reflect.Array;

import java.util.Scanner;

import java.util.regex.\*;

public class Labyrint {

    private Rute[][] ruter;

    public Labyrint(String filnavn) {

        // hvis filnavnet ikke bestaar av 1 eller flere bokstaver + ".in", avslutt programmet

        if (!Pattern.compile("\\w+\\.in").matcher(filnavn).find()) {

            System.out.println("Ugyldig fil format");

            System.exit(0);

        }

        try {

            Scanner leser = new Scanner(new File(filnavn));

            String linje = leser.nextLine();

            int rader = Integer.parseInt(linje.split(" ")[0]);

            int kolonner = Integer.parseInt(linje.split(" ")[1]);

            ruter = new Rute[rader][kolonner];

            int radNr = 0;

            while (leser.hasNextLine()) {

                linje = leser.nextLine();

                for (int kolNr = 0; kolNr < linje.length(); kolNr++) {

                    if (linje.charAt(kolNr) == '#') {ruter[radNr][kolNr] = new SortRute(radNr, kolNr, this); } else { // antar gyldig filformat

                        if (radNr == 0 || radNr == rader - 1 || kolNr == 0 || kolNr == kolonner - 1) {

                          ruter[radNr][kolNr] = new Aapning(radNr, kolNr, this);

                        } else {

                      ruter[radNr][kolNr] = new HvitRute(radNr, kolNr, this);

                        }

                    }

                }

                radNr++;

            }

            for (int i = 0; i < rader; i++) {

                for (int j = 0; j < kolonner; j++) {

                    try {

                        ruter[i][j].settNabo("nord", ruter[i-1][j]);

                    } catch (ArrayIndexOutOfBoundsException e) {} // ikke sett nabo hvis ruten er out of bounds (hvis vi er ved en kant)

                    try {ruter[i][j].settNabo("oest", ruter[i][j+1]);

                    } catch (ArrayIndexOutOfBoundsException e) {}

                    try {ruter[i][j].settNabo("soer", ruter[i+1][j]);

                    } catch (ArrayIndexOutOfBoundsException e) {}

                    try {ruter[i][j].settNabo("vest", ruter[i][j-1]);

                    } catch (ArrayIndexOutOfBoundsException e) {}

                }

            }

            System.out.println("Slik ser labyrinten ut:\n" + this);leser.close();

        } catch (FileNotFoundException e) {e.printStackTrace();}

    }

    public void finnUtveiFra(int rad, int kol) {

        if (rad > ruter.length - 1 || kol > ruter[0].length - 1) {

            System.out.println("Gitt rad eller kolonne finnes ikke i labyrinten");

            return;

        }

        System.out.println(ruter[0].length);

        ruter[rad][kol].finn(ruter[rad][kol]);

    }

}

Rute.java

    @Override

    public void finn(Rute fra) {

        if (naboer.get("nord") != fra && naboer.get("nord") != null) {

            naboer.get("nord").finn(this);

        }

        if (naboer.get("oest") != fra && naboer.get("oest") != null) {

            naboer.get("oest").finn(this);

        }

        if (naboer.get("soer") != fra && naboer.get("soer") != null) {

            naboer.get("soer").finn(this);

        }

        if (naboer.get("vest") != fra && naboer.get("vest") != null) {

            naboer.get("vest").finn(this);

        }

    }

Oblig7 – GUI

Snake.java

import java.awt.\*;

import java.awt.event.\*;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.Queue;

import javax.swing.\*;

public class Snake {

    private String retning = "hoyre";

    private int hodePos = 76;

    private Queue<JLabel> slange = new LinkedList<>();

    private JFrame vindu = new JFrame("Snake - hermagst");

    private JPanel main = new JPanel(new GridBagLayout());

    private JPanel spillPanel = new JPanel(new GridLayout(12, 12, -1, -1));

    public Snake() {

        try {

            UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());

        } catch (Exception e) {

            System.exit(1);

        }

        vindu.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        GridBagConstraints gbc = new GridBagConstraints();

        gbc.gridx = 0;

        gbc.gridy = 0;

        gbc.weightx = 1.0;

        gbc.weighty = 1.0;

        gbc.fill = GridBagConstraints.BOTH;

        JPanel header = new JPanel(new GridBagLayout());

        // ------ KNAPPER FOR AA ENDRE RETNING ------

        class FlyttRetning implements ActionListener {

            String retning;

            public FlyttRetning(String retning) {

                this.retning = retning;

            }

            @Override

            public void actionPerformed(ActionEvent e) {

                endreRetning(retning);

            }

        }

        JButton oppKnapp = new JButton("Opp");

        gbc.gridx = 1;

        header.add(oppKnapp, gbc);

        oppKnapp.addActionListener(new FlyttRetning("opp"));

        JButton venstreKnapp = new JButton("Venstre");

        gbc.gridx = 0; gbc.gridy = 1;

        header.add(venstreKnapp, gbc);

        venstreKnapp.addActionListener(new FlyttRetning("venstre"));

        JButton hoyreKnapp = new JButton("Hoyre");

        gbc.gridx = 2;

        header.add(hoyreKnapp, gbc);

        hoyreKnapp.addActionListener(new FlyttRetning("hoyre"));

        JButton nedKnapp = new JButton("Ned");

        gbc.gridx = 1; gbc.gridy = 2;

        header.add(nedKnapp, gbc);

        nedKnapp.addActionListener(new FlyttRetning("ned"));

        // ------ INITIERING AV RUTENETT ------

        ArrayList<Integer> skatter = new ArrayList<>(); // liste av indekser

        for (int i = 0; i < 10; i++) { // i < 10 for aa starte med 10 skatter

            int tilfeldigInt = (int)Math.floor(Math.random()\*(143-0+1)+0); // 144 ruter aa velge mellom siden 12\*12

            if (skatter.contains(tilfeldigInt) || tilfeldigInt == 76) {

                i -= 1;

                continue; // hvis det tilfeldige tallet allerede har blitt generert, eller tallet er der hvor slangen skal starte, lag et nytt tall

            }

            skatter.add(tilfeldigInt);

        }

        for (int i = 0; i < 144; i++) {

            JLabel boks;

            if (skatter.contains(i)) { // boks har en generert skatt

                boks = new JLabel("$");

                boks.setForeground(Color.red);

                boks.setBackground(Color.white);

            } else if (i == 76) { // der slangen skal starte

                boks = new JLabel("o");

                boks.setBackground(Color.green);

                slange.add(boks);

            } else {

                boks = new JLabel(" ");

                boks.setBackground(Color.white);

            }

            boks.setOpaque(true);

            boks.setFont(new Font("Arial", Font.BOLD, 20));

            boks.setHorizontalAlignment(SwingConstants.CENTER);

            boks.setBorder(BorderFactory.createLineBorder(Color.black));

            spillPanel.add(boks);

        }

        // ------ VISING AV GUI ------

        gbc.gridy = 0;

        header.setPreferredSize(new Dimension(500,100));

        main.add(header, gbc);

        gbc.gridy = 1;

        spillPanel.setPreferredSize(new Dimension(500,500));

        main.add(spillPanel, gbc);

        vindu.add(main);

        vindu.pack();

        vindu.setVisible(true);

        vindu.setFocusable(true);

        vindu.addKeyListener(new FlyttRetningPiltast());

    }

    public void endreRetning(String nyRetning) {

        retning = nyRetning;

    }

    public void flytt(JLabel hode) {

        JLabel nesteBoks;

        if (retning == "opp") {

            if (hodePos <= 11) {

                spillPanel = null; return;

            }

            hodePos -= 12;

            nesteBoks = (JLabel) spillPanel.getComponent(hodePos);

        } else if (retning == "hoyre") {

            if ((hodePos + 1) % 12 == 0) {

                spillPanel = null; return;

            }

            hodePos += 1;

            nesteBoks = (JLabel) spillPanel.getComponent(hodePos);

        } else if (retning == "ned") {

            if (hodePos >= 132) { // hvis hodePos er stoerre enn eller lik foerste rute paa nederste rad...

                spillPanel = null; return;

            }

            hodePos += 12;

            nesteBoks = (JLabel) spillPanel.getComponent(hodePos);

        } else {

            if (hodePos % 12 == 0) {

                spillPanel = null; return;

            }

            hodePos -= 1;

            nesteBoks = (JLabel) spillPanel.getComponent(hodePos);

        }

        if (nesteBoks.getText() == "+") {

            spillPanel = null; return;

        }

        if (nesteBoks.getText().equals("$")) { // hvis neste boks har en skatt...

            slange.offer(nesteBoks);

            for (JLabel slangeBoks : slange) {

                tegnIBoks(slangeBoks, "+", Color.green, Color.black);

            }

            tegnIBoks(nesteBoks, "o", Color.green, Color.black);

            int tilfeldigInt = (int)Math.floor(Math.random()\*(143-0+1)+0);

            JLabel boks = (JLabel)spillPanel.getComponent(tilfeldigInt);

            while (boks.getText() == "o" || boks.getText() == "+" || boks.getText() == "$") { // repeter til generert boks ikke er en del av slangen eller allerede en skatt

                tilfeldigInt = (int)Math.floor(Math.random()\*(143-0+1)+0);

                boks = (JLabel)spillPanel.getComponent(tilfeldigInt);

            }

            tegnIBoks(boks, "$", Color.white, Color.red);

            return;

        }

        slange.offer(nesteBoks);

        for (JLabel slangeBoks : slange) {

            tegnIBoks(slangeBoks, "+", Color.green, Color.black);

        }

        tegnIBoks(nesteBoks, "o", Color.green, Color.black);

        tegnIBoks(slange.remove(), "", Color.white, Color.white);

    }

    public void tegnIBoks(JLabel boks, String text, Color bakgrunn, Color forgrunn) {

        boks.setText(text);

        boks.setBackground(bakgrunn);

        boks.setForeground(forgrunn);

    }

    public JLabel hentHodeRef() {

        return slange.peek();

    }

}