

# **Visualisointikirjaston ylläpitodokumentti**

Viski-ryhmä

Helsinki 31.8.2006

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

**Kurssi**

581260 Ohjelmistotuotantoprojekti (6 ov)

**Projektiryhmä**

Esa Elovaara  
Suvi Hiltunen  
Tomi Jylhä-Ollila  
Riku Louhimo  
Samuli Sairanen  
Juho Vuori

**Asiakas**

CSC / Aleks Kallio

**Johtoryhmä**

Juha Taina  
Jaakko Saaristo

**Kotisivu**

<http://www.cs.helsinki.fi/group/viski>

**Versiohistoria**

Versio	Päiväys	Tehdyt muutokset
1.0	31.8	Dokumentin palautusversio.

# Sisältö

<b>1 Johdanto</b>	<b>1</b>
<b>2 Sanasto</b>	<b>2</b>
<b>3 Arkkitehtuurikäyttäjän käyttöohje</b>	<b>4</b>
3.1 Asennus- ja käynnistysohje . . . . .	4
3.1.1 Tärkeät tiedostot . . . . .	4
3.1.2 Riippuvuudet . . . . .	5
3.1.3 Kirjaston asennus . . . . .	6
3.1.4 Ohjelmakoodin kääntäminen . . . . .	6
3.1.5 Demo-ohjelmien ajaminen . . . . .	6
3.1.6 Yksikkötestien suorittaminen . . . . .	6
3.1.7 Dokumenttien PostScript-versioiden luominen . . . . .	7
3.2 Hierarkkinen puu ja lämpökartta . . . . .	7
3.2.1 Dataset . . . . .	7
3.2.2 Visualisoinnin luominen . . . . .	10
3.2.3 Visualisoinnin muokkaaminen . . . . .	11
3.3 SOM-kartta . . . . .	13
3.3.1 Dataset . . . . .	13
3.3.2 Plot . . . . .	14
3.3.3 Visualisoinnin luominen . . . . .	15
3.4 Visualisointikokoelma . . . . .	16
<b>4 Puutteet ja kehitysehdotukset</b>	<b>17</b>
4.1 Hierarkkinen puu ja lämpökartta . . . . .	17
4.2 SOM-kartta . . . . .	19
4.3 Visualisointikokoelma . . . . .	20
4.4 Koodin ylläpito . . . . .	20
<b>5 Testaus</b>	<b>21</b>
5.1 Yksikkö- ja integrointitestaus . . . . .	21
5.2 Järjestelmätestaus . . . . .	21

# 1 Johdanto

CSC on opetusministeriön omistama tieteen tietotekniikan keskus, joka tarjoaa tutkijoille Suomen laajinta tieteellisten ohjelmistojen ja tietokantojen valikoimaa sekä Suomen tehokkainta superlaskentaympäristöä Funet-tietoliikenneyhteyksien kautta. CSC kehittää korkean tason analyysisovelluksia eri tieteenalojen käyttöön. Yksi konkreettinen esimerkki on bioinformatiikan alaan kuuluva DNA-mikrosirudataa varten kehitetty NAMI-analyysiympäristö. Järjestelmä koostuu graafisesta asiakasohjelmistosta sekä useista palvelinohjelmistoista. Tässä ohjelmistotuotantoprojektissa kehitetään visualisointikirjasto analyysiympäristöä varten.

## 2 Sanasto

**Arkkitehtuurikäyttäjä** Ohjelmoija, joka käyttää visualisointikirjastoa sovelluksessaan.

**Bittikartta** Kuvien tallennusmuoto tietokoneessa. Jokainen kuvan pikseli omaa väritiedon.

**Datajoukko** Joukko datapisteitä.

**Datapiste** Visualisoitavan datan pienin yksikkö, joka voi koostua useasta erillisestä arvosta. Pisteiden arvot voidaan tulkita esimerkiksi kolmiulotteisen avaruuden koordinaateiksi.

**Entiteetti** Visualisoinnin interaktiivinen osa.

**Event** Ohjelmointitekhninen konstruktio, jonka avulla kirjasto kommunikoi sen sisäisistä tapahtumista kirjaston ulkopuolisille ohjelmiston komponenteille.

**Hierarkkinen klusterointi** Datajoukon jakaminen osajoukkoihin ja näiden edelleen jakaminen osiin tietyn yhtäläisen ominaisuuden perusteella.

**Hierarkkinen puu ja lämpökartta** Visualisointi, jolla kuvataan hierarkkisesti klusteroitua dataa kahdessa ulottuvuudessa värityksen avulla. Tätä käytetään esimerkiksi geenianalyyseissa.

**Interaktiivisuus** Interaktiokäyttäjän mahdollisuus vaikuttaa visualisoinnin ulkonäköön.

**Interaktiokäyttäjä** Arkkitehtuurikäyttäjän tekemän sovelluksen käyttäjä.

**JFreeChart** Avoimen lähdekoodin visualisointikirjasto, jolla voidaan piirtää tavallisimpia graafisia esityksiä.

**JPanel** Swing-kirjaston käyttöliittymäkomponentti, johon sijoitetaan muita käyttöliittymäkomponentteja.

**Kirjasto** Tässä dokumentissa sanaa kirjasto käytetään toisinaan viittaamaan tässä toteutettaviin ohjelmakomponentteihin. Tosiasiassa komponentit ovat ainoastaan JFreeChart-kirjaston laajennus.

**Klusteri** Datapisteiden joukko, jonka alkiot muistuttavat toisiaan.

**Kolmiulotteinen hajontakuvi** Kolmiulotteisen pistejoukon projektio kaksiulotteiselle tasolle.

**Kontekstivalikko** Valitun käyttöliittymäkomponentin ja sen osan mukaan muokkautuva valikko.

**MVC-arkkitehtuurimalli** on yleisesti käytetty tapa jakaa sovellus kolmeen erilliseen komponenttiin. Mallikomponentti toimii sovelluksen datavarastona, näkymäkomponentti esittää tiedon graafisesti tai muuten käyttäjälle, ja ohjainkomponentti kontrolloi näitä.

**Kuuntelija** Ohjelmistokomponentti, joka on rekisteröitynyt vastaanottamaan eventejä toiselta komponentilta.

**Pikseli** Näyttö- tai tulostuslaitteen erottelukyvyn yksikkö.

**Solu** SOM-kartan solu, yksittäinen datapiste.

**SOM-kartta** Self-organizing map. Neuroverkkoihin perustuva oppimisalgoritmi. Algoritmia käytetään kaksiulotteisten visualisointien tuottamiseen moniulotteisesta datasta.

**Suljettu klusteri** Hierarkkisen puun visualisoinnissa piilotettu alipuu.

**Swing** Javan graafisten käyttöliittymien luontiin tarkoitettu luokkakirjasto.

**Työkaluvihje** Tekstidialogi, joka kertoo jotain hyödyllistä tietoa hiirisoittimen osoittamasta käyttöliittymäkomponentista.

**Visualisointi** Annetusta datasta tietyllä visualisointimenetelmällä tuotettu kuva.

**Visualisointikokoelma** Yhteen JPanel-käyttöliittymäkomponenttiin liitettyjen visualisointien joukko.

**Visualisointimenetelmä** Algoritmi, jolla numeerinen  $n$ -ulotteinen data muutetaan kaksiulotteiseksi kuvaksi.

## 3 Arkkitehtuurikäyttäjän käyttöohje

Tämä käyttöohje sisältää ohjeet visualisointikirjaston asennukseen, ohjelmakoodin kääntämiseen, demo-ohjelmien ajamiseen ja kirjaston ohjelmointirajapintojen käyttöön. Ohje antaa arkkitehtuurikäyttäjälle perusvalmiudet ottaa käyttöön ja hyödyntää tätä kirjastoa omissa sovelluksissaan.

Asennus- ja käynnistysohje sisältää tietoa kirjaston sisällöstä sekä ohjeet sen käyttöönottoon. Visualisointien ohjelmointirajapintojen käyttö edellyttää visualisoitavan datan järjestämistä kirjaston ymmärtämään muotoon, joten eri visualisointeja käsitteleviin ohjeisiin kannattaa tutustua huolellisesti.

### 3.1 Asennus- ja käynnistysohje

Tämä osio tarjoaa yleiskatsauksen kirjastopakkausten sisältöön sekä antaa ohjeet kirjaston lähdekoodin ja dokumentaation kääntämiseen. Mukana ovat myös ohjeet demo-ohjelmien ajamiseen ja testauksen suorittamiseen.

#### 3.1.1 Tärkeitä tiedostot

Kirjaston julkaisun juurihakemistossa ovat seuraavat hakemistot:

`dist` Sisältää valmiiksi käännetyt jar-pakkaukset sekä dokumenttien PostScript- ja PDF-versiot.

`doc` Sisältää kaiken sisäisen ja ulkoisen dokumentaation Javadoc-kommentteja lukuun ottamatta. Mukana ovat myös LaTeX-lähdekoodit.

`ext` Sisältää visualisointikirjaston tarvitsemia ulkopuolisia kirjastoja.

`extra` Sisältää viski-projektin sivutuotteena syntynyttä koodia. Tämä koodi ei ole virallinen osa visualisointikirjastoa.

`src` Sisältää visualisointikirjaston lähdekoodin (visualisointikomponentit, testaukset ja demo-ohjelmat).

`dist`-hakemiston tärkeät tiedostot ovat:

`dist/viski-1.0.0.jar` Itse visualisointikirjasto.

`dist/hc_demo.jar` Hierarkkisen klusteroinnin demo-ohjelma.

`dist/som_demo.jar` SOM-kartan demo-ohjelma.

`dist/multichart_demo.jar` Visualisointikokoelman demo-ohjelma.

`dist/jcommon-1.0.0.jar` JCommon-kirjasto. Demo-ohjelmien suoritus vaatii tämän kirjaston.

Lisäksi hakemistossa ovat projektin aikana syntyneet dokumentit PostScript- ja PDF-muodoissa.

### 3.1.2 Riippuvuudet

Visualisointikirjaston **ajon aikainen käyttö** on riippuvainen seuraavista komponenteista:

**Java 2 Runtime Environment 1.3** Myöhemmätkin Java-toteutukset voivat toimia, mutta tämä versio on saatavilla osoitteessa <http://java.sun.com/j2se/1.3/>

**JCommon 1.0.0** Sisältää mm. JFreeChart-kirjaston käyttämiä käyttöliittymäkomponentteja.

Pakkaus `viski-1.0.0.jar` sisältää JFreeChart 1.0.1 -kirjaston kaiken toiminnallisuuden ja on osittain ristiriitainen alkuperäisen JFreeChart-kirjaston kanssa. Tämän vuoksi Javan luokkapolussa *ei saa* esiintyä JFreeChart-kirjastoa ajon aikana.

Visualisointikirjaston **ylläpito ja kehitys** on riippuvaista seuraavista komponenteista:

**Java 2 SDK 1.3** Myöhemmätkin Java-toteutukset voivat toimia, mutta tämä versio on saatavilla osoitteessa <http://java.sun.com/j2se/1.3/>

**JFreeChart 1.0.1** Visualisointikirjasto on todellisuudessa laajennus JFreeChart-kirjastolle. JFreeChart 1.0.1 on mukana tämän kirjaston julkaisussa `ext`-hakemistossa.

**JUnit 3.8.1** Yksikkötestausympäristö, jota tarvitaan visualisointikirjaston yksikkötestien suorittamiseen. JUnit on saatavilla osoitteessa <http://www.junit.org/>

**Cobertura 1.8** Cobertura tuottaa yksikkötestauksen kattavuusraportin. Se on saatavilla osoitteessa <http://cobertura.sourceforge.net/>

**ant 1.6.5** Ant-sovellusta tarvitaan lähdekoodin ja Javadoc-dokumentaation kääntämiseen. Se on saatavilla osoitteessa <http://ant.apache.org/>

**LaTeX** LaTeX-ohjelmistoa tarvitaan LaTeX-dokumenttien kääntämiseen PostScript- ja PDF-muotoihin. Nämä dokumentit on testattu pdfTeX:n versiolla 3.141592-1.21a-2.2.

**Make** LaTeX-dokumenttien kääntämiseen on tarjolla Makefile-tiedostot. Näitä on testattu GNU Make:n versiolla 3.80.



### 3.1.3 Kirjaston asennus

Kirjasto asennetaan sijoittamalla `dist`-hakemistosta löytyvät pakkaukset `viski-1.0.0.jar` ja `jcommon-1.0.0.jar` Javan luokkapolkuun (*classpath*). JFreeChart-kirjasto *ei saa* esiintyä luokkapolussa, koska se on ristiriidassa viski-kirjaston kanssa.

### 3.1.4 Ohjelmakoodin kääntäminen

Arkkitehtuurikäyttäjän kirjasto, demo-ohjelmat ja Javadoc-dokumentaatio käännetään ajamalla seuraava komento `src`-hakemistossa:

```
ant
```

Vaihtoehtoisesti komponentit voidaan kääntää yksitellen seuraavasti:

**Kirjasto** `ant compile_lib`

**Hierarkkinen klusterointi** `ant compile_hc`

**SOM-kartta** `ant compile_som`

**Visualisointikokoelma** `ant multichartdemo`

**Javadoc** `ant javadoc`

### 3.1.5 Demo-ohjelmien ajaminen

**Hierarkkisen klusteroinnin** demo-ohjelma suoritetaan `dist`-hakemistossa seuraavalla komennolla:

```
java -jar hc_demo.jar tiedosto
```

Tässä *tiedosto* on klusterointi- ja lämpökarttadatan sisältävä tiedosto, ja näitä tiedostoja löytyy `src/util`-hakemistosta.

**SOM-kartan** demo-ohjelma suoritetaan `dist`-hakemistossa seuraavalla komennolla:

```
java -jar som_demo.jar
```

Tämä luo  $10 \times 10$  -kokoisen kartan esityksen. Ohjelmalle voi myös antaa halutun koon muodossa `java -jar som_demo.jar leveys korkeus`.

**Visualisointikokoelman** demo-ohjelma suoritetaan `dist`-hakemistossa seuraavalla komennolla:

```
java -jar multichart_demo.jar
```

### 3.1.6 Yksikkötestien suorittaminen

Yksikkötestit käännetään ja suoritetaan ajamalla seuraava komento `src`-hakemistossa:

```
ant junit
```

### 3.1.7 Dokumenttien PostScript-versioiden luominen

Dokumentaation LaTeX-koodi käännetään PostScript-dokumenteiksi ajamalla seuraava komento doc-hakemistossa:

```
make
```

Dokumenttien PostScript-versiot tallentuvat dokumenttien omiin alihakemistoihin.

## 3.2 Hierarkkinen puu ja lämpökartta

Tämä osio on arkkitehtuurikäyttäjän käyttöohje hierarkkinen klusterointi ja lämpökartta-visualisoinnin toteuttamiseen. Ohjeen tarkoitus on antaa esimerkkejä kirjaston käytöstä perustilanteissa, joita ovat datan syöttäminen, visualisoinnin muodostaminen sekä visualisoinnin muokkaaminen. Suuri osa tässä esitetyistä koodiesimerkeistä löytyy HCDemo1-ohjelmasta. Demossa on esitelty kokonaisuutena yksi tapa luoda toimiva visualisointi. Kattavat luokka- ja metodikuvaukset löytyvät API-dokumentaatiosta.

### 3.2.1 Dataset

Datan syöttämiseen käytetään seuraavia pakkauksen `org.jfree.data.hc` luokkia:

- `org.jfree.data.hc.HCDataSet` extends `org.jfree.data.general.AbstractDataset`
- `org.jfree.data.hc.HeatMap` extends `org.jfree.data.xy.MatrixSeries`
- `org.jfree.data.hc.HCTreeNode`
- `org.jfree.data.hc.DataRange`

**HeatMap** Luokassa säilytetään lämpökartan datajoukkoa, sekä rivien ja sarakkeiden nimitietoja. Luokka perii JFreeChart-kirjastosta luokan `org.jfree.data.xy.MatrixSeries`.

Esimerkki uuden HeatMap-olion luomisesta ja rivien ja sarakkeiden nimien syöttämisestä:

```
BufferedReader input =
    new BufferedReader(
        new InputStreamReader(
            new FileInputStream("filename")));

//name of the HeatMap
String name = "nameOfThisHeatMap";
```

```

//number of the rows
int height = 100;
//number of the columns
int width = 100;

//Constructs a new heatmap
HeatMap heatMap = new HeatMap (name, height, width);
try {
    for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
            String line = input.readLine();
            //Updates the value of the specified item
            // in this HeatMap.
            heatMap.update(y,x,Double.parseDouble(line));
        }
    }
} catch (Exception e) {}

//Sets the names of the columns
for (int x = 0; x < width; x++) {
    heatMap.setColumnName(x, "ColumnName"+x);
}
//Sets the names of the rows
for (int y = 0; y < height; y++) {
    heatMap.setRowName(y, "RowName"+y);
}

```

**HCTreeNode** Luokka ylläpitää tietoa yksittäisestä klusteripuun solmusta. Siihen tallennetaan tieto solmun korkeudesta, sekä viitteet sen vanhempaan ja molempiin lapsiin. Jokaisella HCTreeNode-oliolla voi olla joko kaksi tai ei yhtään lasta. Uusi, haarasolmua esittävä HCTreeNode-olio luodaan näin:

```

//height of the node
double height = 2;
HCTreeNode node = new HCTreeNode(height);

```

Uuden lehtisolmua esittävän HCTreeNode-olion luomisen yhteydessä asetetaan sen `DataRange`, joka esittää kokonaislukuväliä `[index, index]`, missä `index` on sen lämpökartan rivin tai sarakkeen numero, johon kyseinen solmu liittyy. Uusien lehtisolmujen luominen:

```

//height of the node
double height = 1;
//the heatmap row or column
int leftIndex = 0;

```

```
//the heatmap row or column
int rightIndex=1;
```

```
HCTreeNode leftChild = new HCTreeNode(height, leftIndex);
HCTreeNode rightChild = new HCTreeNode(height, rightIndex);
```

Kun lehtisolmu liitetään haarasolmun lapseksi, päivittyy haarasolmun DataRange kokonaislukuväliksi  $[leftIndex, rightIndex]$ . Lapsien asettaminen:

```
node.setLeftChild(leftChild);
node.setRightChild(rightChild);
```

Jos klusteripuun data on tallennettu binääripuuna, voi sen solmuista luoda HCTreeNode-oliot seuraavankaltaisen rekursiivisen metodin avulla:

```
//reads the height of the node from the input data
BufferedReader input;
//the leftmost row and column index of the heatmap
int index = 0;

HCTreeNode leftTree;
HCTreeNode topTree;

try {
    input =
        new BufferedReader(
            new InputStreamReader(
                new FileInputStream("filename")));

    //a root node of the left clustering tree
    leftTree = readTree(input,index);

    //a root node of the top clustering tree
    topTree = readTree(input,index);
} catch (Exception e){}
.
.
.
private static HCTreeNode readTree(BufferedReader input,int index)
    throws Exception {

    String line;
    HCTreeNode node;
    HCTreeNode leftChild;
    HCTreeNode rightChild;
```

```

line = input.readLine();

//char "l" in the input data represents a leaf node
if (line.equals("l")) return new HCTreeNode(0,index);

//a branch node
node = new HCTreeNode(Double.parseDouble(line));

leftChild = readTree(input, index);
node.setLeftChild(leftChild);

rightChild = readTree(
    input, node.getDataRange().getRightBound()+1);
node.setRightChild(rightChild);

return node;
}

```

Kun HCDataset-olio luodaan, se kutsuu HCTreeNode-luokan metodia `finalizeTree()`. Tämän jälkeen HCTreeNode-olioita ei voi enää muuttaa.

**HCDataset** Luokassa kootaan lämpökartan ja hierarkkisten puiden esittämiseen tarvittava data. Tämä tehdään kutsumalla konstruktoria, jolle annetaan parametrina lämpökartan ja molempien puiden datajoukot. Lämpökartan datajoukolle ei voi antaa arvoa null. Sen sijaan puun datajoukon voi jättää antamatta asettamalla sen arvoksi null. Luokka perii JFreeChart-kirjastosta luokan `org.jfree.data.general.AbstractDataset`. Uuden HCDataset-olion voi luoda seuraavasti:

```
HCDataset dataset = new HCDataset(heatmap, leftTree, topTree);
```

### 3.2.2 Visualisoinnin luominen

Hierarkkisen klusteroinnin ja lämpökartan piirtämiseen käytetään pakkauksen `org.jfree.chart.plot` luokkaa `HCPlot`. Se piirtää HCDataset-olion määrittelemät datajoukot. Uuden HCPlot-olion luominen:

```
HCPlot plot = new HCPlot(dataset);
```

Uuden JFreeChart-olion voi konstruoida pakkaudesta `org.jfree.chart.plot` löytyvän `BioChartFactory`-luokan avulla.

```
JFreeChart chart = BioChartFactory.createHCChart(
```

```

        "Hierarchical Clustering Demo", // chart title
        dataset,                        // data
        true,                          // tooltips
        false                           // URLs
    );

```

Valmis visualisointi voidaan saada aikaan myös luomalla HCPlot-olio, joka sitten sijoitetaan JFreeChart-olioon seuraavasti:

```

HCPlot plot = new HCPlot(dataset);
JFreeChart chart =
    new JFreeChart("Title",           //title
        JFreeChart.DEFAULT_TITLE_FONT, //font
        plot,                         //plot
        false                          //createLegend
    );

```

Jälkimmäisessä tapauksessa on asetettava myös työkaluvihjeiden luomiseen käytettävä HCToolTipGenerator-olio. Lämpökartan visualisoinnissa työkaluvihjeitä käytetään ruutuja vastaavien datapisteiden arvojen näyttämiseen, ja klusteripuiden visualisoinneissa klusterin koon havainnollistamiseen. Metodin kutsuminen null-arvolla poistaa työkaluvihjeet käytöstä.

```
plot.setToolTipGenerator(new StandardHCToolTipGenerator());
```

Jos JFreeChart-oliota ei ole luotu käyttäen BioChartFactory-luokkaa, kannattaa asettaa anti-aliasointi pois päältä, jotta visualisointi piirtyisi nopeammin ja siistimmin.

```
chart.setAntiAlias(false);
```

### 3.2.3 Visualisoinnin muokkaaminen

Plot-luokan metodien avulla arkkitehtuurikäyttäjä voi muokata visualisoinnin ulkonäköä haluamaansa muotoon.

Oletusarvoisesti molemmat puut ovat näkyvissä, jos ne ovat olemassa. Puut voidaan piilottaa seuraavasti:

```

plot.hideRowTree();
plot.hideColumnTree();

```

Puut voidaan asettaa uudestaan näkyviin:

```

plot.showRowTree();
plot.showColumnTree();

```

Rivien ja sarakkeiden nimet näytetään oletusvisualisoinnissa. Arkkitehtuurikäyttäjän valinnan mukaan nekin voidaan piilottaa tai asettaa uudelleen näkyviin:

```
plot.showColumnNames();
plot.showRowNames();
plot.hideColumnNames();
plot.hideRowNames();
```

Jos interaktiokäyttäjä valitsee lämpökartan solun tai hierarkkisen puun solmun visualisoinnissa, valintaa vastaavat lämpökartan solut korostetaan valkoisella reunuksella. Arkkitehtuurikäyttäjä voi asettaa tämän ominaisuuden pois päältä seuraavasti:

```
plot.setSelectionHighlight(false);
```

Suljettua klusteria vastaava lämpökartan rivi tai sarake korostetaan mustalla reunuksella. Tämän ominaisuuden saa pois käytöstä seuraavasti:

```
plot.setAverageHighlight(false);
```

Puiden esittämiseen varattua tilaa voidaan muuttaa. Metodeille annetaan parametrina puun koko, joka on osuus visualisoinnin näkyvästä osasta.

```
double size = 0.2;
plot.setRowTreeSize(size);
plot.setColumnTreeSize(size);
```

Rivien ja sarakkeiden nimien esittämiseen varattua tilaa muutetaan vastaavasti:

```
double size = 0.15;
plot.setRowNamesSize(size);
plot.setColumnNamesSize(size);
```

Lämpökartan värityksen muuttamista varten on ensin luotava uusi GradientColorPalette-olio, jonka konstruktorille annetaan parametrina uusi väritys Map-rajapinnan toteuttavana oliona.

```
// A mapping between values of type double and
// a color to be used as key colors
Map keyColorMap = new TreeMap();
keyColorMap.put((Object)new Double(-1.0), new Color(255,0,0));
keyColorMap.put((Object)new Double(0.0), new Color(0,0,255));
keyColorMap.put((Object)new Double(1.0), new Color(0,255,0));

GradientColorPalette color =
    new GradientColorPalette(keyColorMap);

//to change the coloring
plot.setColoring(color);
```

Olemassaolevan värityksen voi vaihtaa logaritmiseksi. Tätä varten pitää ensin selvittää jo käytössä oleva `GradientColorPalette`-olio.

```
GradientColorPalette palette;
palette = plot.getColorizing();

//logarithmic value mapping
palette.setLinear(false);

//linear value mapping
palette.setLinear(true);
```

Arkkitehtuurikäyttäjä voi avata ja sulkea kerralla kaikki visualisoinnissa näkyvät puiden haarat. Tarvittava metodi löytyy `HCTreeNodeInfo`-luokasta.

```
AbstractHCClusteringInfo columnInfo;
AbstractHCClusteringInfo rowInfo;

columnInfo = plot.getColumnClusteringInfo();
rowInfo = plot.getRowClusteringInfo();

//to expand all
columnInfo.getRoot().setSubTreeOpen(true);
rowInfo.getRoot().setSubTreeOpen(true);

//to collapse all
columnInfo.getRoot().setSubTreeOpen(false);
rowInfo.getRoot().setSubTreeOpen(false);
```

### 3.3 SOM-kartta

Tämä osio on arkkitehtuurikäyttäjän käyttöohje SOM-karttojen visualisoinnin toteuttamiseen. Ohjeen tarkoitus on antaa esimerkkejä kirjaston käytöstä perustilanteissa, joita ovat datan syöttäminen, visualisoinnin muodostaminen ja visualisoinnin muokkaaminen. Kattavat luokka- ja metodikuvaukset löytyvät API-dokumentaatiosta.

#### 3.3.1 Dataset

Datan syöttämiseen käytetään seuraavia pakkauksen `org.jfree.data.som` luokkia:

- `org.jfree.data.som.SOMDataset` extends `org.jfree.data.general.AbstractDataset`
- `org.jfree.data.som.SOMDataItem`



**SOMDataset** Luokassa kootaan SOM-kartan soluja esittävä data. Tämä tehdään kutsuamalla konstruktoria, jolle annetaan parametreina kartan korkeus ja leveys. Tämän jälkeen SOMDataset-luokkaan sijoitetaan SOMDataItem-olioita, joista jokainen vastaa yhtä SOM-kartan solua. Solun arvo ei voi olla null, eli kartassa ei voi olla tyhjiä soluja.

**SOMDataItem** Yksi solu sisältää seuraavat tiedot:

- Solun värin,
- Solun kuvauksen/kuvaukset,
- Solun neuronipainoarvot.

Esimerkki koodista, joka luo SOMDataItem-olion:

```
SOMDataItem item
= new SOMDataItem(Color.red,
                    new String[] {new String("A"), new String("B")},
                    new double[] {2.0, 0.9});
```

Esimerkki koodista, joka luo SOMDataset-olion ja täyttää sen SOMDataItem-olioilla:

```
int columns = 10;
int rows = 15;

SOMDataset set = new SOMDataset(columns, rows);

for (int i=0; i < rows; ++i) {
    for (int j=0; j < columns; ++j) {
        String[] s = new String[] {"A", "B"};
        double[] w = {Math.random(), Math.random(), Math.random()};
        float r = (float)Math.random();
        float g = (float)Math.random();
        float b = (float)Math.random();
        Color clr = new Color(r,g,b);
        SOMDataItem sdi = new SOMDataItem(clr, s, w);
        dataset.addValue(j, i, sdi);
    }
}
```

### 3.3.2 Plot

SOM-kartan piirtämiseen käytetään pakkauksen `org.jfree.chart.plot` luokkaa:

- `org.jfree.chart.plot.SOMPlot` extends `Plot`

**SOMPlot** SOMPlot piirtää SOMDataset-olion määrittelemän kartan. Arkkitehtuuri-käyttäjä voi vaikuttaa kartan ulkonäköön kutsumalla seuraavia SOMPlot-olion metodeja:

```
setDescriptionFont(Font descriptionFont)
```

Asettaa solukuvauksissa käytetyn kirjasimen.

```
setToolTipGenerator(SOMToolTipGenerator toolTipGenerator)
```

Asettaa työkaluvihjeiden luomiseen käytetyn olion. SOMToolTipGenerator luo SOMDataItem-luokan sisältämästä datasta merkkijonon, jota käytetään työkaluvihjeenä. Metodin kutsuminen null-arvolla poistaa työkaluvihjeet käytöstä.

Esimerkki koodista, joka luo SOMPlot-olion ja muuttaa sen asetuksia:

```
SOMDataset dataset;
.
.
.
SOMPlot plot = new SOMPlot(dataset);

plot.setToolTipGenerator(new StandardSOMToolTipGenerator());
plot.setDescriptionFont(new Font("Monospaced", Font.PLAIN, 8));
```

### 3.3.3 Visualisoinnin luominen

Valmis visualisointi saadaan aikaan luomalla SOMPlot-olio, joka sitten sijoitetaan JFreeChart-olioon.

```
SOMDataItem dataset;
.
.
.
SOMPlot plot = new SOMPlot(dataset);
JFreeChart chart = new JFreeChart("Title",
                                   JFreeChart.DEFAULT_TITLE_FONT,
                                   plot,
                                   false
                                   );
```

BioChartFactory-luokan metodi

```
JFreeChart createSOMChart(String title,
                           SOMDataset dataset,
                           boolean tooltips,
                           boolean urls)
```

yhdistää edellä mainitut vaiheet.

### 3.4 Visualisointikokoelma

**MultiChartPanel** Luokka toimii kuten `ChartPanel`-luokka. Erona on, että jokaisen konstruktorin loppuun on lisätty `List`-tyyppinen parametri `panels`, joka sisältää viittauksen listaan, jossa ovat `MultiChartPanel`-oliot, jotka halutaan tulostaa tai tallentaa samanaikaisesti tämän paneelin tulostus/tallennus-toimintoa käyttäen. Kun `MultiChartPanel`-olioita luodaan, oletetaan, että niistä jokainen lisätään tähän listaan. Listan ylläpito hoidetaan pääohjelmassa eli se on arkkitehtuurikäyttäjän vastuulla. `MultiChartPanel`-luokasta ei löydy `Get/set`-metodia, joten lista oletetaan luoduksi ennen kuin oliot luodaan ja lisätään sinne. Näiden metodien lisääminen tosin olisi triviaalia, jos sille löytyy tarvetta. Parametri `panels` voi olla `null`, jolloin `MultiChartPanel` käyttäytyy kuin normaali `ChartPanel`.

Ainut poikkeuksellinen konstruktori on `MultiChartPanel(list, chartpanel)`, jossa olemassa olevasta `ChartPanel`-oliosta tehdään `MultiChartPanel`. Tämä on kuitenkin ongelmallinen siinä mielessä, että `ChartPanel`-luokan `save`-, `print`- ja `zoom`-valintoja kontekstimenussa ei voida siirtää uuteen paneeliin, vaan uudessa `MultiChartPanel`-oliossa ne ovat oletuksena päällä.

Sekä tallennettaessa että tulostettaessa käytetään komennon saavan `MultiChartPanel`-komponentin korkeutta ja leveyttä kaikkien listassa olevien paneelien sisältämien visualisointien kokona. Eri kokoiset visualisoinnit tallentuvat ja tulostuvat ruudukkona, samankokoisina ruutuina.

Tallennus- ja tulostustoiminnoissa on määritelty muuttuja `saveColumns`, joka määrittää kuinka monta visualisointia halutaan vierekkäin tulosteisiin tai tallennettaviin kuviin. Tämä on kovakoodattu lukumäärään 2, mutta se voidaan muuttaa. Jos haluttaisiin, voitaisiin tästä tehdä luokkamuuttuja ja kirjoittaa `get/set`-metodit sitä varten. Tallennustoiminnot sijaitsevat `MultiChartUtilities`-luokassa, ja tulostustoiminto `MultiChartPanel`-luokassa, mutta sama edellä mainittu muuttuja löytyy molemmista. Tulostukseeseen liittyy erikoistapaus, jossa visualisointien määrä on pienempi tai yhtäsuuri kuin `saveColumns`-muuttuja ja tulostus toteutetaan portrait-muodossa paperille, mikä aiheuttaa normaalisti visualisointien liiallisen skaalautumisen pystysuunnassa. Tämä on "purkkakorjattu" lisäämällä tarkistus, joka aiheuttaa ylimääräisen tallennusruudukorivin lisäyksen.

**MultiChartUtilities** Luokka on kirjastoluokka, jossa on visualisointien tallentamista varten vastaavia julkisia metodeita kuin `ChartUtilities`-luokassa. Nä-

mä kuitenkin ovat tehty käytettäviksi vain `MultiChartPanel`-luokan yhteydessä. `MultiChartPanel`-luokassa on oletuksena PNG-tallennus, kuten `ChartPanel`-luokassakin. JPEG-tallennuksen saa toimimaan muuttamalla `MultiChartPanel`-luokan `doSaveAs()` metodin png:tä käsittävät kohdat jpeg:in vastaaviksi.

**MultiChartSaveDemo** Luokka esittelee yksinkertaisesti, miten `MultiChartPanel`-luokkaa käytetään. Ensin luodaan tyhjä lista. Sitten luodaan `MultiChartPanel`-olio, jonka konstruktorissa viitataan tyhjään listaan. Tämän jälkeen juuri luotu `MultiChartPanel` lisätään listaan. Tätä toistetaan jokaisen `MultiChartPanel`-olion kohdalla, jotka halutaan tulostuvan/tallentuvan samaan kohteeseen. Seuraavassa koodia kyseisestä demo-luokasta:

```
//THE LIST, which has the bookkeeping of MultiChartPanels
public java.util.List panels;
panels = new ArrayList();

//new panel with some chart
MultiChartPanel panel1 = new MultiChartPanel(chart, panels);
panels.add(panel1);

//other panel
MultiChartPanel panel2 = new MultiChartPanel(chart2, panels);
panels.add(panel2);
```

## 4 Puutteet ja kehitysehdotukset

Tässä luvussa esitellään toimitettavan tuotteen tunnetut puutteet ja lisäksi muita kehitysehdotuksia. Osa ehdotuksista on tuotteen selkeästi keskeneräisiä ominaisuuksia, osa projektin loppuvaiheessa syntyneitä ajatuksia siitä, miten tuotetta voisi parantaa, mutta joita ei ehditty edes aloittaa toteuttaa.

### 4.1 Hierarkkinen puu ja lämpökartta

Hierarkkisen puun ja lämpökartan kannalta suurin ongelma on `JFreeChart`in `ChartPanel`-luokka. Se ei ole lainkaan käyttökelpoinen interaktiivisten sovellusten tekoon. Siitä pitäisi kirjoittaa versio, joka paremmin soveltuu interaktiiviseen käyttöön ja `HCPPlot`-luokka pitäisi sovittaa tähän. Tämä on interaktiivisen käytön kannalta ratkaisevaa, sillä nykyisellään `ChartPanel` pakottaa näytön täydellisen päivityksen aivan liian herkästi, joka tekee kirjastosta hyvin hitaan suurten datajoukkojen käsittelyssä.

Avattaessa tai suljettaessa klusterointipuun solmua lämpökartta piirretään uudelleen sille varatun alueen vasempaan ylänurkkaan. Interaktiivisessa käytössä olisi hyvä, jos sol-

mun aukaiseminen tai sulkeminen klikkaamalla sitä ei aiheuttaisi kyseisen solmun paikan muuttumista. Tätä ei kokonaisuudessaan voi toteuttaa JFreeChart-kirjaston sisällä, sillä esimerkiksi visualisoinnin vierittäminen toteutetaan sen ulkopuolella. Tätä ei nykyisellä ohjelmointirajapinnalla kuitenkaan voi toteuttaa edes sen ulkopuolelta. Hyödyllinen muutos olisi kehittää edelleen `ClusteringTreeChangeEvent`-luokkaa siten, että eventin mukana toimitetaan käyttäjälle tieto kyseisen solmun sijainnista. Tämä mahdollistaisi toiminnon toteuttamisen.

Hierarkkisen puun ja lämpökartan visualisoinnissa klusterointipuut piirretään tavallisesti lämpökartan ylä- ja vasemmalle puolelle, rivien ja sarakkeiden nimet taas piirretään lämpökartan ala- ja oikealle puolelle. Itse piirtorutiinit toteutettiin kuitenkin niin, että klusterointipuiden sekä rivien ja sarakkeiden sijainti voisi olla käänteinen. Ominaisuuden käyttöönottoon ei ole kuitenkaan minkäänlaista ohjelmointirajapintaa. Piirtojärjestystä kontrolloivat oletustoteutuksessa `HPlot`-luokan attribuutit `drawFromLeftToRight` ja `drawFromTopToBottom`. Yksinkertaisimmillaan näiden attribuuttien muokkaamiseen pitäisi kirjoittaa rajapinnat.

`GradientColorPalette`-luokkaan aloitettiin logaritmisen värityksen toteuttaminen. Ominaisuus ei kuitenkaan toimi oikein. Luokan metodi `getColor` pitäisi korjata tältä osin.

Tällä hetkellä osa visualisoinnin geometriaan liittyvästä laskennasta tehdään `HPlotState`-luokassa, osa `HPlot`-luokan metodissa `draw` ja sen apumetodeissa. Tämä pitäisi selkeyttää mieluiten niin, että kaikki geometriaan liittyvä laskenta siirretään `HPlotState`-luokkaan.

Lämpökartan väritykseen liittyvä toiminnallisuus pitäisi uudistaa. Tällä hetkellä `HPlot`-luokan metodit `setColoring` ja `getColoring` manipuloivat `GradientColorPalette`-olioita. Tätä varten pitäisi kuitenkin luoda abstrakti yläluokka `AbstractColorPalette`, josta `GradientColorPalette` perittäisiin. Näin väritys voitaisiin tarvittaessa toteuttaa kokonaan erityyppisellä palettioliolla. Tämä muutos vaatisi myös `setColorPaletteEditor`- ja `getColorPaletteEditor`-metodien toteuttamisen, joilla voisi asettaa `HPlot`-luokan `paletteEditor`-attribuutin. Näin voitaisiin toteuttaa myös uusi käyttöliittymä värien muokkaamiseen.

Tällä hetkellä kaikki lämpökartan pisteiden valintaan liittyvä tieto säilytetään `Rectangle`-tyyppisessä oliossa. Parempi tapa olisi säilyttää tietoa `Set`-oliossa. Tämän jälkeen olisi mahdollista myös toteuttaa monimutkaisempien pistejoukkojen valintaan liittyvät toiminnot.

JFreeChartin `CategoryAxis`-luokka soveltuu vain tilanteisiin, joissa datajoukko jakautuu "kategorioihin" vain toisen akselinsa suuntaisesti. Tämä on turhan yksinkertaistava oletus, eikä toteudu lämpökartan yhteydessä. Ongelma saatiin ratkaistua `HCMediator`-apuluokalla. Tämä luokka pitäisi poistaa ja sen sijaan korjaus pitäisi tehdä sinne, minne se kuuluu: `CategoryAxis`-luokkaan. Tämä ei kuitenkaan ole aivan yksinkertaista toteuttaa siististi, joten sitä ei tehty viski-projektin yhteydessä.

JFreeChartin rakennetta ajatellen klusterointipuut olisi siisteintä toteuttaa erillisenä `ClusteringTreeAxis`-luokkana, joka toteuttaisi samantapaisen rajapinnan kuin

CategoryAxis. Tällöin klusterointipuita voisi periaatteessa käyttää myös muiden JFreeChart-visualisointien yhteydessä. Tällöin visualisointi koostuisi selkeästi kolmesta erityyppisestä osiosta: klusteripuista, lämpökartoista ja rivien- sekä sarakkeiden nimistä.

ClusteringTreeAxis-ajatusta seuraten toteuttamatta jäänyt lämpökartan sarakkeiden tai rivien värikoodipalkki olisi syytä toteuttaa myös samantapaisena oliona. Tämän toteuttaminen olisi helppoa hyvin samanlaisen rajapinnan avulla, millä CategoryAxis on tällä hetkellä tehty.

## 4.2 SOM-kartta

```
public List getArea(SOMDataItem item1, SOMDataItem item2)
```

Molempien SOMDataItem-olioiden paikka matriisissa selvitetään aikavaativuudeltaan luokkaa  $O(n)$  olevalla funktiolla. Olisi tehokkaampaa jos getArea()-metodia kutsuva luokka pitäisi yllä tietoa item1 ja item2 olioiden paikasta matriisissa, jolloin getArea() voisi ottaa parametreinä olioiden indeksit matriisissa.

```
public JPanel getPanel()
public String getPanelName()
```

SOMPlot-luokan asetukset ja toiminnot sisältävän paneelin luominen tulisi siirtää SOMPlotEditor-nimiseen luokkaan, joka olisi periytetty PlotEditor-luokasta. Viski-projektin perustana käytetty JFreeChart-kirjaston versio ei vielä tue erikoistuneita PlotEditor-luokkia, mutta koodin siirtäminen olisi silti järkevää, koska se poistaisi SOMPlot-luokasta sellaista koodia, joka ei suoranaisesti liity SOM-karttojen piirtämiseen.

```
private Color contrastingColor(Color color)
```

Solun kuvaustekstin ja valintaindikaattorin värin määrittely olisi järkevää siirtää omaan SOMColorScheme-luokkaansa, joka annettaisiin SOMPlot-olion konstruktorille parametrina. SOMColorScheme-oliolla tulisi olla viite sen omistavaan SOMPlot-olioon, jotta se pystyisi selvittämään SOM-kartan sen hetkisen taustavärin. SOMColorScheme-luokan ja siitä periytetyn luokan tulisi sisältää seuraavat metodit:

```
Color contrastingColorForDescription(Color color)
```

Metodi määrittelee kuvaustekstin värin.

```
Color contrastingColorForSelection(Color color)
```

Metodi määrittelee valintaindikaattorin värin.

```
void drawSOM()
```

Metodissa käytettävät `fillPolygon`- ja `drawPolygon`-metodit eivät piirrä polygonien ääri viivoja täysin samoihin kohtiin, josta seuraa se, että solujen valintaindikaattorit saattavat piirtyä katselijaa häiritsevällä tavalla. Itsetehdyt piirtometodit saattaisivat ratkaista ongelman.

Metodin piirtämät kuusikulmiot eivät aina ole symmetrisiä. Tämä johtuu ilmeisesti pyöristysvirheistä.

```
private void drawDescriptions()
```

Metodi sisältää bugikorjauksen, jonka takia solun kuvausteksti piirretään kahdesti. Jos näin ei tehdä, vasemman yläkulman solun kuvausteksti piirtyy kapeammalle alueelle kuin muiden solujen kuvaustekstit. Tämän virheen korjaamiseen löytyy luultavasti jokin siistimpi tapa, jonka voisi löytää tutkimalla tarkemmin `TextBlock`-luokkaa.

Kuvaustekstien piirtäminen `TextBlock`-luokkaa käyttäen on SOM-kartan piirtämiseen käytettävistä operaatioista työläin. Jos tekstit jätetään piirtämättä, kartan piirtämiseen käytetty aika putoaa murto-osaan. Jos kuvaustekstien piirtäminen olisi tehokkaampaa, pystyisi `SOMPlot` käsittelemään kertaluokkaa suurempia karttoja.

### 4.3 Visualisointikokoelma

Visualisointikokoelma voisi tulevaisuudessa ottaa huomioon jokaisen paneelin koon erikseen ja pyrkiä asemoimaan visualisoinnit samassa järjestyksessä, jossa ne ovat ruudulla. Tämä olisi todennäköisesti mahdollista pyytämällä `MultiChartPanel`-olioita asemoivalta `JFrame`-oliolta tai vastaavalta `getComponents()` metodin kautta tietoja eri komponenteista, joita graafisesta käyttöliittymästä löytyy. Näiden komponenttien joukosta poimittaisiin `MultiChartPanel`-oliot ja niiden sijaintien ja kokojen mukaan annettaisiin asemointi tulostettavaan tai tallennettavaan kuvaan.

### 4.4 Koodin ylläpito

Kirjaston jatkokehittämistä ajatellen sen ensisijainen dokumentointi on kirjaston Javadoc-komentointi. Tämän lisäksi hyödyllistä tietoa löytyy myös kirjaston suunnittelu- ja muutosdokumenteista.

Keskeinen ja todennäköisimmin vastaan tuleva jatkokehitystarve on viski-kirjaston soveltaminen `JFreeChart`in uusiin versioihin. On mahdotonta ennakoida `JFreeChart`in tulevaisuudessa tehtäviä muutoksia, mutta mikäli sen rakenne ei ratkaisevasti muutu, ei soveltamisen pitäisi olla vaikeaa. Lähes kaikki viski-projektin tuottama koodi on `JFreeChart`ista erillisinä tiedostoinaan irrallaan muusta `JFreeChart`ista. Poikkeuksen tähän tekee `DefaultPlotEditor`, johon oli `JFreeChart`in arkkitehtuuristen puutteiden vuoksi pakko tehdä muutoksia. Nämä muutokset on pakko siirtää käsin seuraavaan versioon.

Luokka on dokumentoimaton ja se on selkeästi suunniteltu toteutettavaksi nykyisestä poikkeavalla tavalla. On siis todennäköistä, että JFreeChartin tulevissa versioissa tämä luokka saattaa muuttua ratkaisevasti. Luokka toteuttaa visualisointien Properties-ikkunan, joka on muusta koodista irrallinen toiminto, joten muutosten siirtäminen luokan uusiin versioihin tuskin on kovin suuri urakka.

Lopullinen tavoite olisi viski-kirjaston integroiminen osaksi normaalia JFreeChartia. Tämä helpottaisi päivitysongelmia sekä ainakin HCMediator-luokkaan liittyvien ongelmien ratkaisemista.

## 5 Testaus

Testausta toteutettiin testaussuunnitelman pohjalta.

### 5.1 Yksikkö- ja integrointitestausta

Yksikkötestauksen apuna käytettiin Cobertura-ohjelmistoa. Testaussuunnitelmassa määritelty yksikkötestauksen vaatimus rivikattavuuden suhteen tuli täytetyksi. Coberturan tuottama kattavuusraportti käytiin läpi, ja rivit, joihin testit eivät koskaan päässeet olivat paikkoja joihin oli mahdotonta päästä.

Ainut poikkeus tähän sääntöön on MultiChartPanel-luokka, jota testattiin enemminkin järjestelmätestauksen pohjalta, sillä `print()` ja `doSaveAs()` metodien toiminnallisuuden testausta on vaikea ellei mahdoton automatisoida.

Integrointitestausta suoritettiin yksikkötestauksen ohessa, koska yksikkötestien kirjoittaminen vaati muidenkin luokkien käyttämistä. Näin ollen yksikkö- sekä integrointitestausta sulautettiin yhteen. Nämä testit kirjoitettiin JUnit-kirjastoa apuna käyttäen. Yksikkötesteissä pyrittiin arvoalueanalyysiä käyttäen löytämään virheitä luokkien toiminnallisuuksista.

JUnit-testit jokaiselle luokalle löytyvät `junit/-` alihakemistosta. Cobertura-ohjelmiston tuottama html-raportti löytyy hakemistosta `doc/coverage/`.

### 5.2 Järjestelmätestaus

Järjestelmätestaus toteutettiin käyttämällä valmiita visualisointeja ja käymällä läpi vaatimusdokumenttiin kirjattuja kohtia. Samalla varmistuttiin ohjelmiston toimivuudesta ja siitä, että ohjelma on toteutettu kuten se oli vaatimusmäärittelyssä määritelty.

Nämä vaatimusmäärittelyssä olevat kohdat jäivät toteuttamatta osaltaan tai kokonaan:

8. Kirjasto mahdollistaa visualisointien yhdistämisen samaan JPaneliin.

- Visualisointeja ei yhdistetä samaan JPaneliin vaan JPaneleista ylläpidetään tietoa, jota käyttämällä voidaan toteuttaa useiden JPaneleiden tallentaminen ja tulostaminen samaan



kohteeseen.

21. Visualisointiin voidaan lisätä väripalkki puun ja lämpökartan väliin. Palkissa kunkin rivin tai sarakkeen kohdalle on merkitty värikoodi. Väri ilmoittaa, mihin ryhmään kyseinen rivi tai sarake kuuluu.

- Visualisointiin ei toteutettu väripalkkia puun ja lämpökartan väliin.

65. Visualisointiin voidaan lisätä väripalkki puun ja lämpökartan väliin. Palkissa kunkin rivin tai sarakkeen kohdalle on merkitty värikoodi. Väri ilmoittaa, mihin ryhmään kyseinen rivi tai sarake kuuluu.

- Visualisointiin ei toteutettu väripalkkia puun ja lämpökartan väliin.

66. Väripalkkiin liittyvät ryhmätiedot annetaan kirjastolle ohjelmointirajapinnan kautta.

- Visualisointiin ei toteutettu väripalkkia puun ja lämpökartan väliin.

39. Tuotetut visualisoinnit voidaan tallentaa tiedostoon JFreeChartin jo toteutetun tallennustoiminnon avulla. Visualisoinnit voidaan tallentaa PNG- ja EPS-muodossa.

- Ei toteutunut täysin, EPS-muotoista tallennusta ei toteutettu.

49. Datan esityssuunta voidaan vaihtaa siten, että määritelty vaaka-akseli muutetaan pystyakseliksi ja pystyakseli vaaka-akseliksi (transpoosi).

- Transpoosia ei toteutettu, mutta koodissa on valmius transpoosin piirtämiseen.

62. Puun visualisoinnissa viemä tila suhteessa lämpökartan viemään tilaan on arkkitehtuurikäyttäjän määriteltävissä. Kirjasto osaa itse varautua tilanteisiin, joissa puuta yritetään määritellä liian suureksi tai pieneksi.

- Kirjasto ei osaa varautua tilanteisiin, joissa puuta yritetään määritellä liian suureksi tai pieneksi.

71. Kirjasto määrittää kartalle minimikoon, jota pienemmäksi sitä ei voi skaalata.

- Tämä on `ChartPanel`-luokan määrittämä asia, joten sitä ei ole toteutettu.

Kolmiulotteisen hajontakuvion osioita (3., 26-31., 74-85.) ei toteutettu.