

# **Suunnitteludokumentti**

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
0.0	1.6	Runko lisätty SVN-varastoon
0.1	13.7	SOM-kartta hyvässä kunnossa, hierarkkinen puu valmistuu hissukseen
1.0	25.7	Tarkastusta varten jäädytetty versio.
1.1	2.8	Lopullinen suunnitteludokumentin jäädytetty versio.

# Sisältö

<b>1 Johdanto</b>	<b>1</b>
<b>2 Sanasto</b>	<b>2</b>
<b>3 Järjestelmäarkkitehtuuri</b>	<b>4</b>
<b>4 Hierarkkinen puu ja lämpökartta</b>	<b>4</b>
4.1 Käyttöliittymä . . . . .	7
4.2 Ohjelmointirajapinnat . . . . .	13
4.2.1 DataRange . . . . .	13
4.2.2 HCDataset . . . . .	14
4.2.3 HCTreeNode . . . . .	14
4.2.4 HeatMap . . . . .	16
4.2.5 HCPlot . . . . .	16
4.2.6 HCNodeInfo . . . . .	21
4.2.7 HCNodeInfoRoot . . . . .	22
4.2.8 HCTreeNodeEntity . . . . .	22
4.2.9 HeatMapBlockEntity . . . . .	23
4.2.10 GradientColorPalette . . . . .	23
4.2.11 HCMediator . . . . .	24
4.2.12 HCToolTipGenerator . . . . .	25
4.2.13 StandardHCToolTipGenerator . . . . .	25
4.3 Komponentin toiminta . . . . .	25
4.3.1 HCPlot . . . . .	25
4.3.2 HCTreeNodeEntity . . . . .	29
<b>5 SOM-kartta</b>	<b>30</b>
5.1 Käyttöliittymä . . . . .	30
5.2 Ohjelmointirajapinnat . . . . .	36
5.2.1 SOMDataItem . . . . .	36
5.2.2 SOMDataset . . . . .	37
5.2.3 SOMPlot . . . . .	38
5.2.4 SOMToolTipGenerator . . . . .	39

5.2.5	StandardSOMToolTipGenerator . . . . .	39
5.2.6	SOMItemEntity . . . . .	40
5.3	Komponentin toiminta . . . . .	41
<b>6</b>	<b>Visualisointikokoelma</b>	<b>42</b>
6.1	Ohjelmointirajapinnat . . . . .	42
6.1.1	MultiChartPanel . . . . .	42
6.1.2	MultiChartUtilities . . . . .	43
6.2	Komponentin toiminta . . . . .	44
<b>7</b>	<b>Kolmiulotteinen hajontakuvio</b>	<b>47</b>

# 1 Johdanto

Viski on ohjelmistotuotantoprojektiryhmä Helsingin yliopiston tietojenkäsittelytieteen laitoksella kesällä 2006. Ryhmän tehtävänä on laajentaa Java-pohjaista visualisointikirjastoa bioinformatiikan tarpeita vastaamaan.

Projektin tuotosta käytetään CSC:n NAMI-projektissa, mutta sen on tarkoitus olla myös yleiskäyttöinen tieteellinen visualisointikirjasto. Työ toteutetaan jo olemassa olevan JFreeChart-kirjaston laajennuksena. Kirjasto tarjoaa rajapinnat erilaisten tietolähteiden syöttämiseen ja toteuttaa kolme visualisointia: hierarkkisen puun ja lämpökartan, SOM-kartan ja kolmiulotteisen hajontakuvion, joista viimeinen suunnitellaan ja toteutetaan vain mikäli projektin aikataulu antaa myöten.

Tämä suunnitteludokumentti on ryhmän sisäinen suunnitelma, jonka perusteella varsinainen toteutus tapahtuu. Kaikki suunnittelun päättymisen jälkeen tehtävät muutokset, jotka toteutusvaiheessa mahdollisesti havaitaan tarpeellisiksi, tullaan lisäämään muutosdokumenttiin. Hajontakuvion osalta mahdolliset suunnitelmat kuitenkin lisätään tähän dokumenttiin.

## 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** Ohjelmointitekniikka, 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 geenianalyysissä.

**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 hajontakuva** 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.

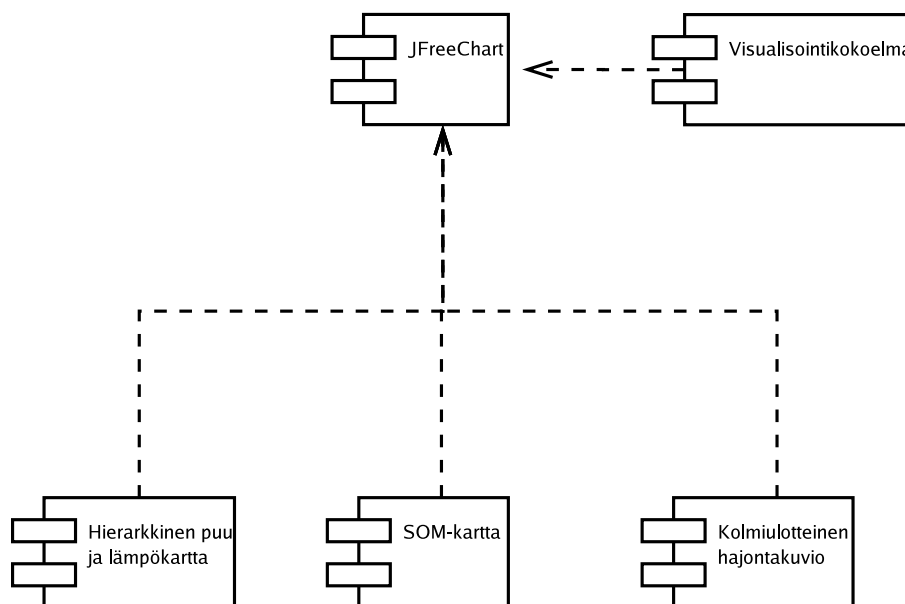
**Visualisointi** Annetusta datasta tietyllä visualisointimenetelmällä tuotettu kuva.

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

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

### 3 Järjestelmäarkkitehtuuri

Visualisointikirjasto toteutetaan JFreeChart-kirjaston laajennuksena. JFreeChart jakaantuu pääpiirteissään datan käsittelyyn liittyvään koodiin ja itse visualisointien toteutukseen liittyvään koodiin. Samaa jakoa noudatetaan myös näiden visualisointien kohdalla. Järjestelmä koostuu kolmesta lähes irrallisesta komponentista: hierarkkisesta puusta ja lämpökartasta, SOM-kartasta sekä kolmiulotteisesta hajontakuviosta. Näistä jokainen käyttää JFreeChart-kirjaston jo toteutettuja ominaisuuksia, mutta näiden välistä kommunikaatiota ei ole. Komponenteilla on myös itsenäiset ohjelmointirajapinnat. Rajapinnat kuitenkin pyritään suunnittelemaan toistensa ja JFreeChart-kirjaston olemassaolevien rajapintojen kanssa mahdollisimman yhtenäisiksi. Kirjaston arkkitehtuuri on kuvattu yleisellä tasolla kuvassa 1.



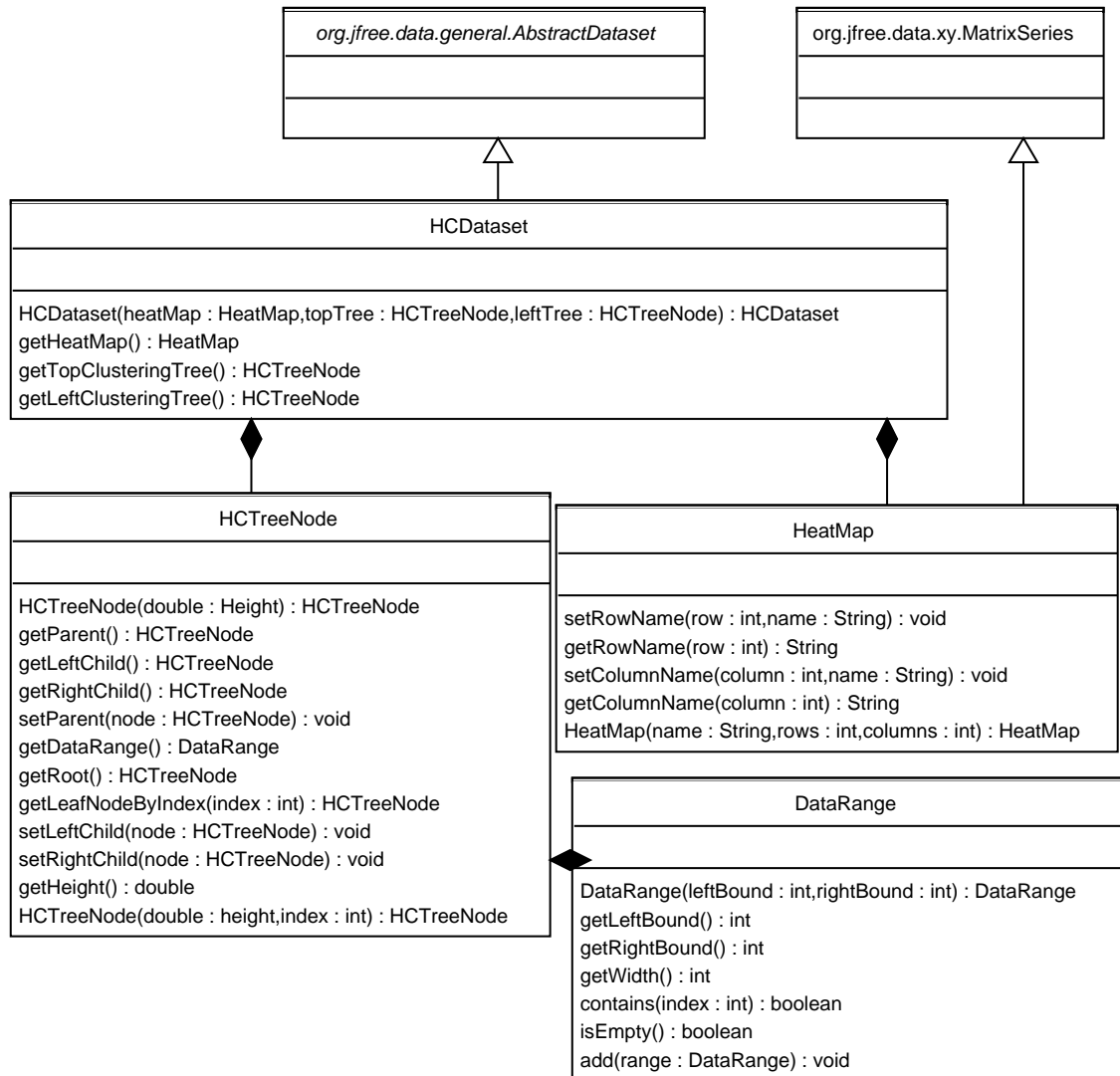
Kuva 1: Visualisointikirjaston arkkitehtuurin yleiskuva.

Kunkin visualisointikomponentin arkkitehtuuri noudattelee MVC-arkkitehtuurimallia (Model-View-Controller). *HCDataset* ja *SOMDataset*, sekä näihin liittyvät apuluokat toimivat visualisointien mallina. *HCPLOT* ja *SOMPlot* sekä näihin liittyvät apuluokat toteuttavat visualisointien näkymän. Ohjainkomponentti on osin toteutettu kirjaston *Plot*-luokkien sisällä, mutta suunnittelun lähtökohtana on toteuttaa näkymän ja mallin rajapinta siten, että visualisoinnin ohjaimen voi toteuttaa myös kokonaan kirjaston ulkopuolella.

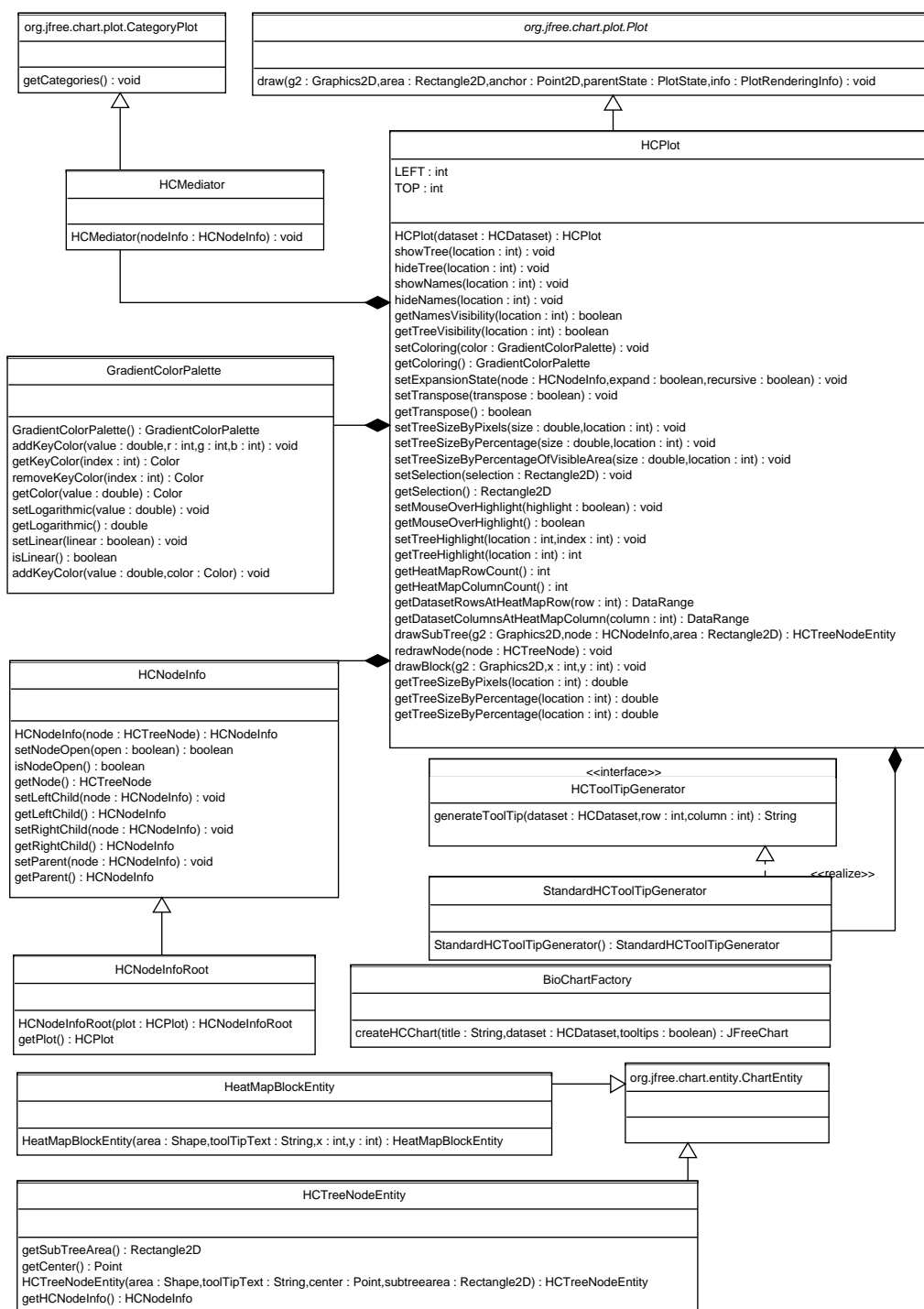
### 4 Hierarkkinen puu ja lämpökartta

Hierarkkisen puun ja lämpökartan toteutus jakaantuu karkeasti kahteen osaan: datan esitykseen ja itse visualisointiin. Datan esitykseen liittyvien kirjaston osien luokkakaavio on esitetty kuvassa 2. Graafisen osion luokkakaavio on esitetty kuvassa 3.





Kuva 2: Hierarkkisen puun ja lämpökartan data-osion luokkakaavio.



Kuva 3: Hierarkkisen puun ja lämpökartan graafisen osion luokkakaavio.

## 4.1 Käyttöliittymä

Kuva 4 on esimerkki hierarkkisen klusteroinnin visualisoinnista perustilassa. Hierarkkiset puut näkyvät lämpökartan vasemmalla ja yläpuolella. Sarakkeiden nimet esitetään alhaalla ja rivien nimet oikealla, mikäli rivikorkeus ja sarakkeleveys riittävät tekstin esittämiseen. Kuvan 4 tilanteessa rivikorkeus on liian matala, joten rivien nimiä ei näytetä.

Jos kohdistin siirretään hierarkkisen puun solmun kohdalle, solmua vastaava klusteri korostetaan kuvan 5 mukaisesti arkkitehtuurikäyttäjän valinnan mukaan.

Jos solmua napsautetaan hiiren kakkospainikkeella, näytetään kontekstivalikko, jossa on yleisten JFreeChart-luokkien toimintojen lisäksi klusterin sulkemiseen ja avaamiseen liittyviä toimintoja (kuva 6). Mahdollisia toimintoja ovat:

- “Collapse”, jolla suljetaan tämä klusteri;
- “Collapse All”, jolla suljetaan koko puu;
- “Expand”, jolla avataan tämä klusteri;
- “Expand Subtree”, jolla avataan tämä klusteri ja kaikki aliklusterit; sekä
- “Expand All”, jolla avataan koko puu.

Solmun napsauttaminen hiiren ykköspainikkeella valitsee kyseistä solmua vastaavan klusterin. Valittu klusteri näytetään lämpökartalla valkoisella reunuksella kuten kuvassa 7.

Solmun kaksoisnapsautus sulkee tai avaa kyseisen klusterin ja vastaa kontekstivalikon toimintoja “Collapse” ja “Expand”. Kuva 8 esittää suljettua (ja valittua) klusteria. Lämpökartalla näytetään suljetun klusterin kohdalla mittausarvojen keskiarvot.

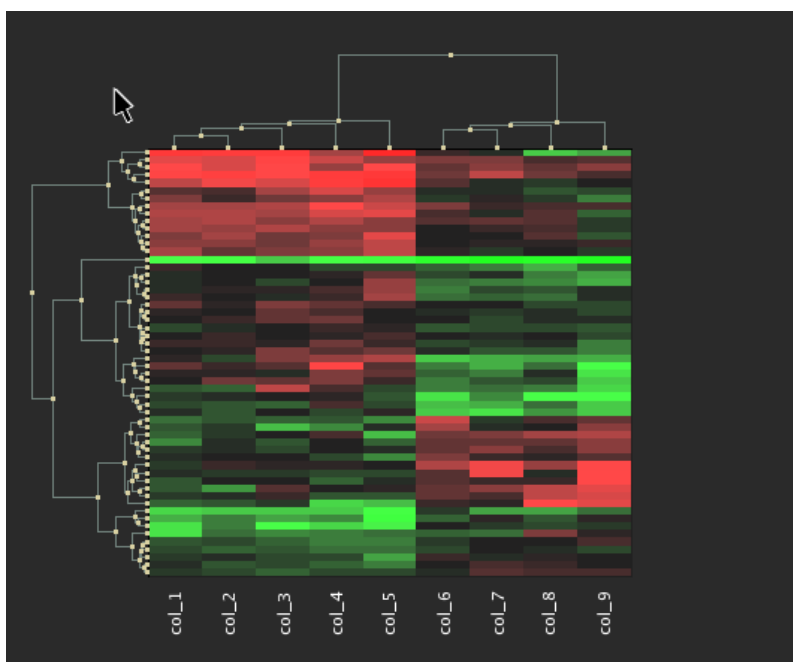
Kuvassa 9 esitetään kuvan 8 tilanne suurennettuna. Tässä kuvassa rivien nimet näkyvät, koska rivikorkeus on siihen riittävä. Suljetun klusterin kohdalla näytetään klusterin ensimmäisen ja viimeisen rivin nimet erotettuna kolmella pisteellä.

Jos kohdistin siirretään lämpökartalle, arkkitehtuurikäyttäjän valinnan mukaan voidaan näyttää mittausarvo työkaluvihjeessä kuvan 10 mukaisesti. Datapisteen sijoittuminen klusteroinnissa voidaan visualisoida puissa.

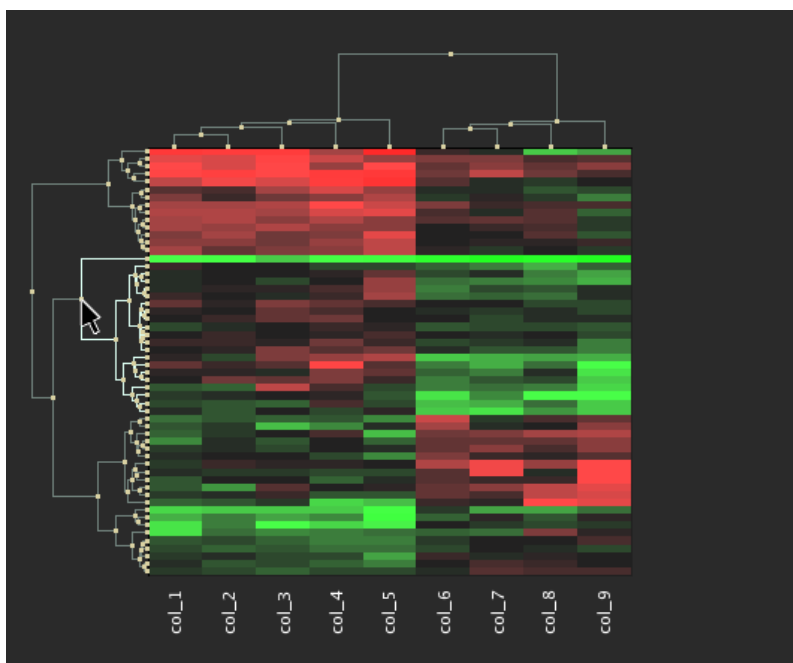
Datapisteen napsauttaminen hiiren ykköspainikkeella valitsee kyseisen mittausarvon, mikä visualisoidaan ympäröimällä datapiste valkoisella reunuksella (kuva 11).

Lämpökartan napsautus hiiren kakkospainikkeella avaa kontekstivalikon, joka ei sisällä tähän visualisointiin liittyviä erikoistoimintoja (kuva 12). Kontekstivalikon kohta “Properties...” avaa valikon, jossa on joitakin toimintoja tämän visualisoinnin muuttamiseen.

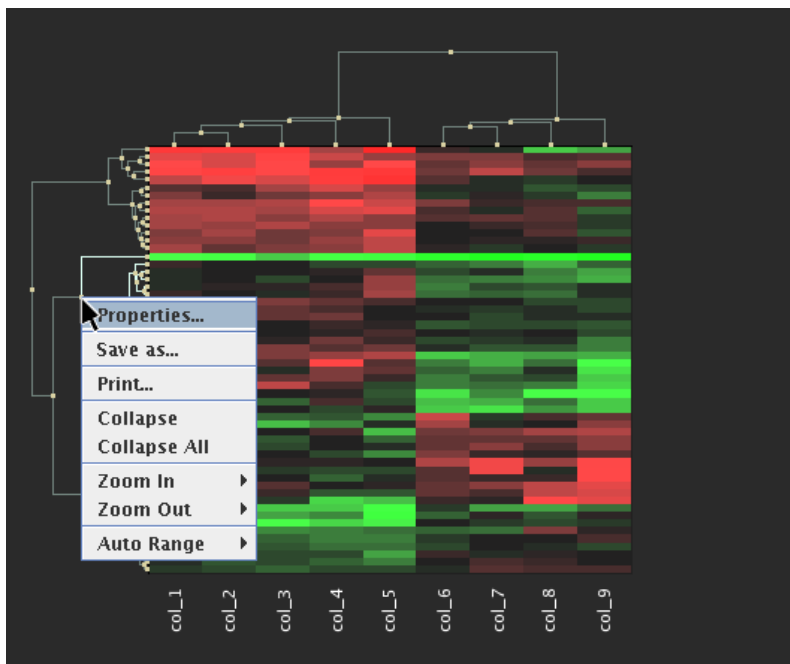
Kuva 13 havainnollistaa Properties-valikkoon kuuluvia toimintoja. Hierarkkisten puiden, rivien nimien ja sarakkeiden nimien visualisointi voidaan ottaa pois päältä sekä datapisteiden värivalikoimaa voidaan muuttaa.



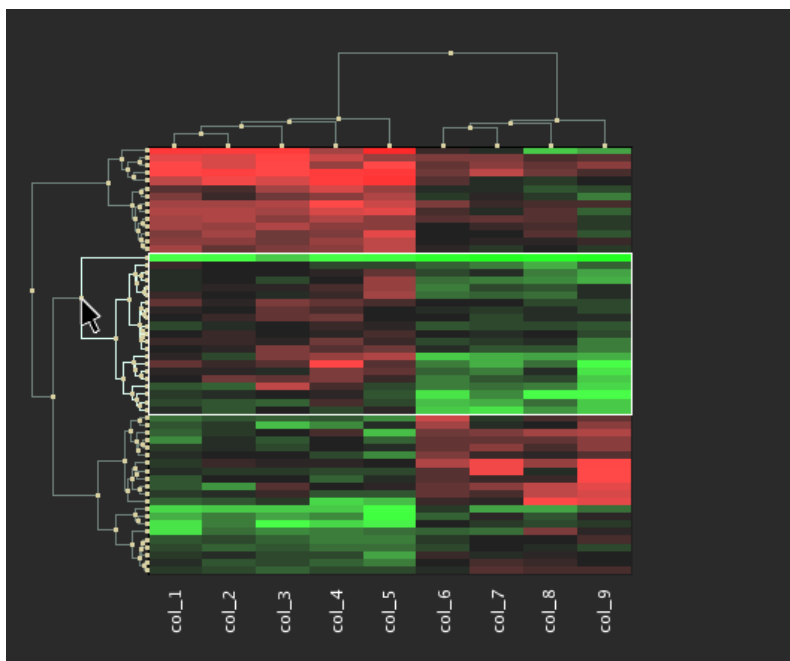
Kuva 4: Hierarkkinen klusterointi perustilassa.



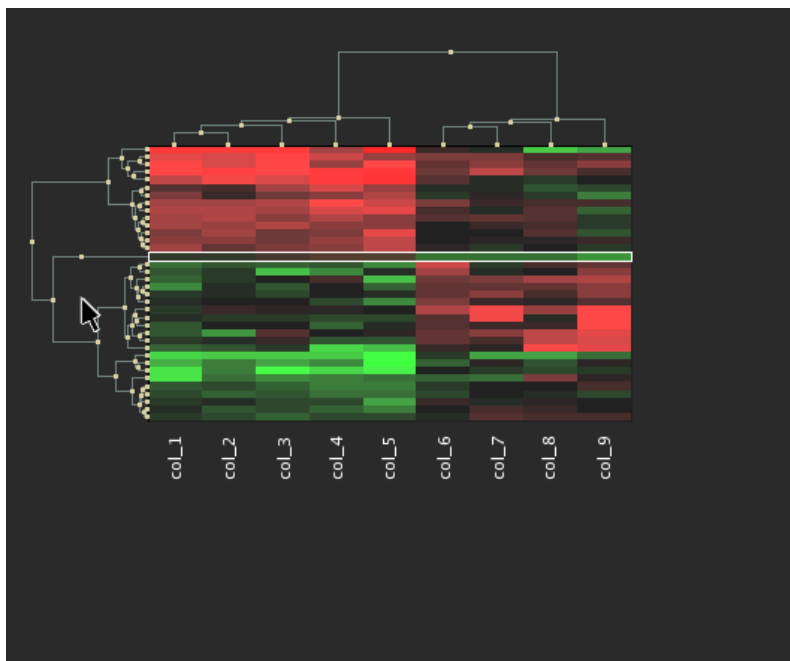
Kuva 5: Kohdistin hierarkkisen puun solmun kohdalla.



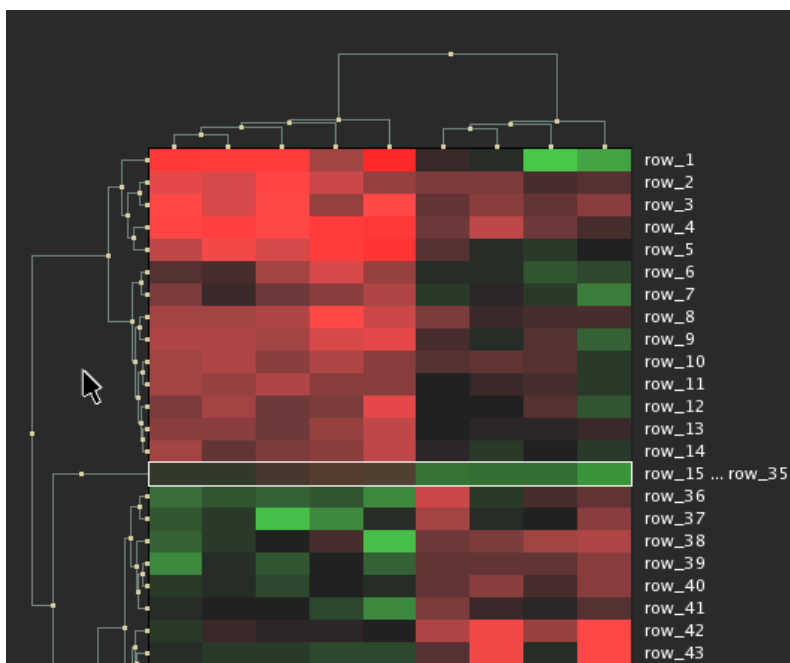
Kuva 6: Hierarkkisen puun solmun kontekstivalikko.



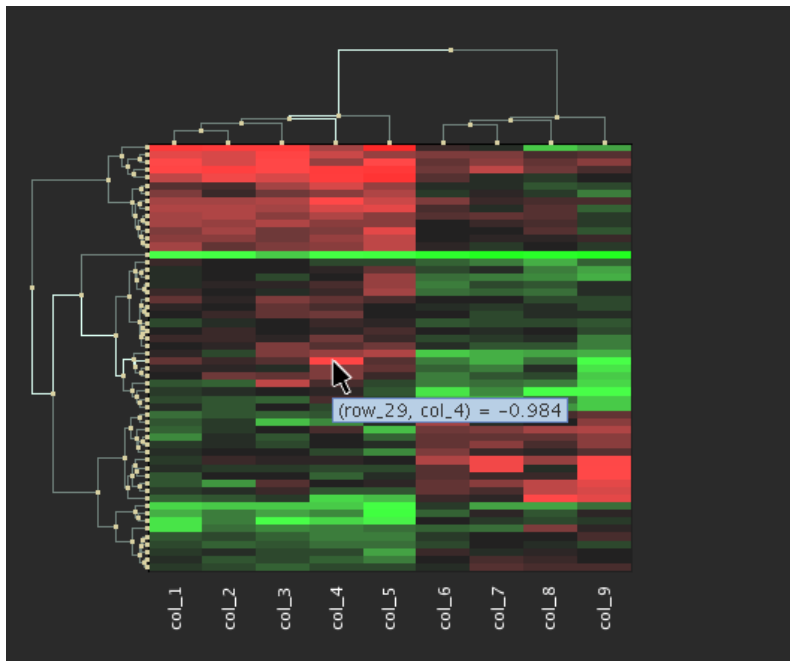
Kuva 7: Hierarkkisen klusterin valinta.



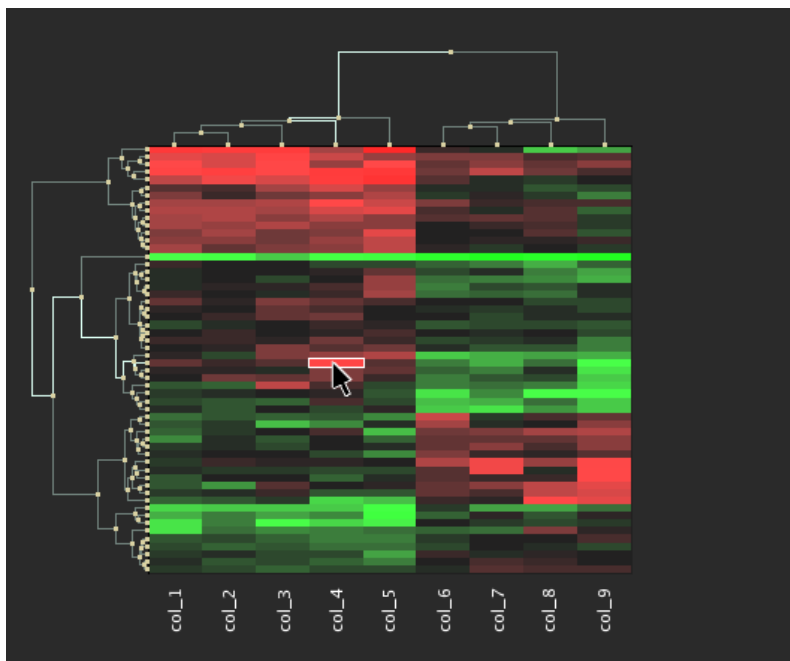
Kuva 8: Suljettu (ja valittu) klusteri.



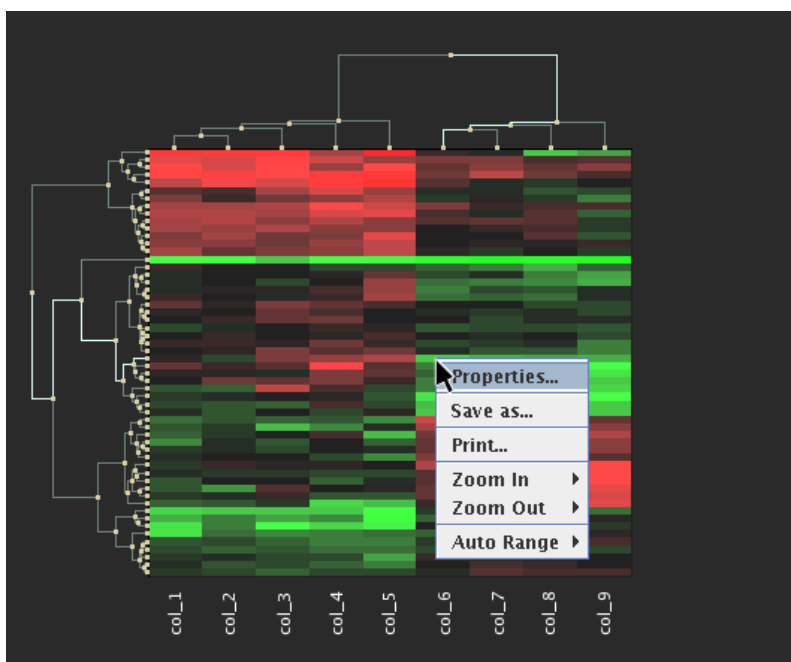
Kuva 9: Pystysuunnassa suurennettu esitys, jossa on suljettu klusteri.



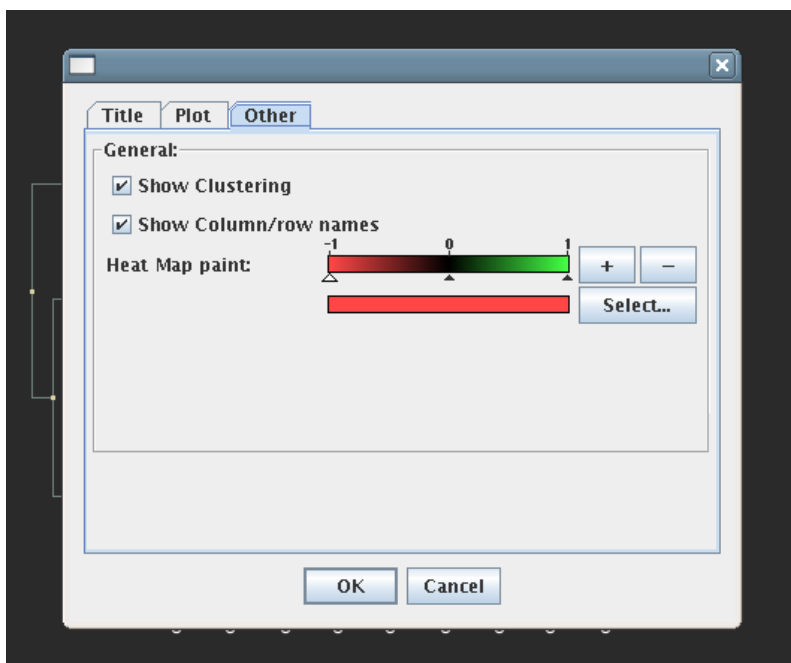
Kuva 10: Kohdistin lämpökartalla.



Kuva 11: Datapisteen valinta lämpökartalla.



Kuva 12: Kontekstivalikko lämpökartalla.



Kuva 13: Hierarkkisen klusteroinnin Properties-valikko.



## 4.2 Ohjelmointirajapinnat

Tässä luvussa kuvataan kaikkien visualisointiin liittyvien luokkien ohjelmointirajapinnat. Rajapinnoista on kuvattu vain ne osat, jotka on tehty tätä visualisointia varten. Jo dokumentoituja JFreeChart-kirjaston rajapintoja ei dokumentoida uudestaan.

### 4.2.1 DataRange

Luokka esittää kokonaislukuväliä. Puun jokainen lehtisolmu liittyy johonkin lämpökarttamatriisiin riviin tai sarakkeeseen, joka on sen *DataRange*. Haarasolmun *DataRange* muodostuu sen allaolevien lehtisolmujen etäisyydestä.

Luokan konstruktori on:

```
public DataRange(int leftBound, int rightBound)
```

Oliot kuvaavat väliä  $[leftBound, rightBound]$ . Tyhjää väliä kuvaavat oliot, joissa  $rightBound < leftBound$ .

Luokka toteuttaa seuraavat julkiset metodit:

```
public int getLeftBound()
```

Palauttaa vasemman rajan indeksin.

```
public int getRightBound()
```

Palauttaa oikean rajan indeksin.

```
public int getWidth()
```

Palauttaa välin leveyden.

```
public boolean contains(int index)
```

Palauttaa *true*, jos ja vain jos *index* kuuluu väliin.

```
public boolean isEmpty()
```

Palauttaa *true*, jos ja vain jos väli on tyhjä.

```
public void add(DataRange range)
    throws DataRangeMismatchException
```

Lisää väliin toisen välin *range*. Heittää poikkeuksen *DataRangeMismatchException*, jos ja vain jos välien väliin jää tyhjää.

### 4.2.2 HCDataset

*extends AbstractDataset*

HCDataset sisältää hierarkkisten puiden ja lämpökartan tietosisällön.

Luokan konstruktori on:

```
HCDataset(HeatMap heatMap,
           HCTreeNode topTree,
           HCTreeNode leftTree)
           throws NullPointerException
```

Asettaa lämpökartan *heatMap* ja molemmat klusteripuut. Konstruktori heittää poikkeuksen *NullPointerException*, jos ja vain jos *heatMap = null*.

Luokka toteuttaa seuraavat julkiset metodit:

```
HCTreeNode getTopClusteringTree()
```

Palauttaa vaakaklusteroinnin datajoukon. Jos joukkoa ei ole, metodi palauttaa arvon *null*.

```
HCTreeNode getLeftClusteringTree()
```

Palauttaa pystyklusteroinnin datajoukon. Jos joukkoa ei ole, metodi palauttaa arvon *null*.

```
HeatMap getHeatMap()
```

Palauttaa lämpökartan datajoukon.

### 4.2.3 HCTreeNode

HCTreeNode on klusteripuun yksittäisen solmun toteutus.

Luokan konstruktorit ovat:

```
HCTreeNode(double height)
```

Asettaa haarasolmulle korkeuden *height*.

```
HCTreeNode(double height, int index)
```

Asettaa lehtisolmulle korkeuden *height* ja asettaa *DataRange*-välin [*index*, *index*].

Luokka toteuttaa seuraavat metodit:

```
void setParent(HCTreeNode node)
    throws NotAChildException
```

Asettaa solmun vanhemmaksi solmun *node*. Metodi heittää poikkeuksen *NotAChildException*, jos ja vain jos tämä solmu ei ole solmun *node* lapsi.

```
HCTreeNode getParent()
```

Palauttaa solmun vanhemman.

```
void setLeftChild(HCTreeNode node)
    throws DataRangeMismatchException
```

Asettaa solmun vasemmaksi lapseksi solmun *node*. Metodi heittää poikkeuksen *DataRangeMismatchException*, jos ja vain jos lapsen *DataRange* ei ole yhteensopiva.

```
HCTreeNode getLeftChild()
```

Palauttaa solmun vasemman lapsen.

```
void setRightChild(HCTreeNode node)
    throws DataRangeMismatchException
```

Asettaa solmun oikeaksi lapseksi solmun *node*. Metodi heittää poikkeuksen *DataRangeMismatchException*, jos ja vain jos lapsen *DataRange* ei ole yhteensopiva.

```
HCTreeNode getRightChild()
```

Palauttaa solmun oikean lapsen.

```
HCTreeNode getRoot()
```

Palauttaa puun juurisolmun.

```
HCTreeNode getLeafNodeByIndex(int index)
    throws IndexOutOfBoundsException
```

Palauttaa lehtisolmun, joka vastaa klusteroinnin riviä *index*. Metodi heittää poikkeuksen *IndexOutOfBoundsException*, jos ja vain jos klusteroinnissa ei ole riviä *index*.

```
DataRange getDataRange()
```

Palauttaa solmun kaikkien alipuitten lehtisolmujen arvovälin.

#### 4.2.4 HeatMap

*extends MatrixSeries*

Luokka sisältää lämpökartan tietosisällön ja rivien ja sarakkeitten nimet.

Luokan konstruktori on:

```
HeatMap(String name, int rows, int columns)
```

Asetetaan uusi matriisi sarakemäärällä *columns* ja rivimäärällä *rows*.

Luokka toteuttaa seuraavat julkiset metodit:

```
void setRowName(int row, String name)
    throws IndexOutOfBoundsException
```

Asettaa rivin *row* nimeksi *name*. Metodi heittää poikkeuksen *IndexOutOfBoundsException*, jos ja vain jos matriisin datajoukossa ei ole riviä *row*.

```
String getRowName(int row)
    throws IndexOutOfBoundsException
```

Palauttaa rivin *row* nimen. Metodi heittää poikkeuksen *IndexOutOfBoundsException*, jos ja vain jos matriisin datajoukossa ei ole riviä *row*.

```
void setColumnName(int column, String name)
    throws IndexOutOfBoundsException
```

Asettaa sarakkeen *column* nimeksi *name*. Metodi heittää poikkeuksen *IndexOutOfBoundsException*, jos ja vain jos matriisin datajoukossa ei ole saraketta *column*.

```
String getColumnName(int column)
    throws IndexOutOfBoundsException
```

Palauttaa sarakkeen *column* nimen. Metodi heittää poikkeuksen *IndexOutOfBoundsException*, jos ja vain jos matriisin datajoukossa ei ole saraketta *column*.

#### 4.2.5 HCPlot

*extends Plot*

Luokan avulla piirretään visualisoinnin ulkoinen käyttöliittymä.

Luokan konstruktori on:

```
HCPlot(HCDataset dataset)
```

Asettaa kentän *dataset*.

Luokkaan liittyvät metodit ovat:

```
void showTree(int location)
    throws NonexistentTreeException
```

Näyttää puun sijainnissa *location*. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos sijainnissa *location* ei ole puuta.

```
void hideTree(int location)
    throws NonexistentTreeException
```

Piilottaa puun sijainnissa *location*. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos sijainnissa *location* ei ole puuta.

```
void showNames(int location)
```

Näyttää rivien tai sarakkeiden nimet sijainnissa *location*.

```
void hideNames(int location)
```

Piilottaa rivien tai sarakkeiden nimet sijainnissa *location*.

```
boolean getTreeVisibility(int location)
    throws NonexistentTreeException
```

Palauttaa tiedon siitä, onko puu näkyvissä sijainnissa *location*. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos sijainnissa *location* ei ole puuta.

```
boolean getNamesVisibility(int location)
```

Palauttaa tiedon siitä, ovatko nimet näkyvissä sijainnissa *location*.

```
void setTranspose(boolean transpose)
```

Asettaa visualisoinnin transpoosin, eli vaakaklusterit esitetään pystyklustereina ja päinvastoin, jos ja vain jos *transpose = true*.

```
boolean getTranspose()
```

Palauttaa tiedon siitä, onko visualisointi transponoitu.

```
void setTreeSizeByPixels(double size, int location)
    throws NonexistentTreeException, InvalidSizeException
```

Varaa tilaa sijainnissa *location* olevalle puulle *size* pikseliä visualisoinnissa. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos puita ei ole olemassa, ja poikkeuksen *InvalidSizeException*, jos ja vain jos puun koko yritetään laittaa liian suureksi tai pieneksi.

```
void setTreeSizeByPercentage(double size, int location)
    throws NonexistentTreeException, InvalidSizeException
```

Varaa tilaa sijainnissa *location* olevalle puulle *size* prosenttia visualisoinnin kokonaisalueesta. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos puita ei ole olemassa, ja poikkeuksen *InvalidSizeException*, jos ja vain jos puun koko yritetään laittaa liian suureksi tai pieneksi.

```
void setTreeSizeByPercentageOfVisibleArea(double size,
                                           int location)
    throws NonexistentTreeException, InvalidSizeException
```

Varaa tilaa sijainnissa *location* olevalle puulle *size* prosenttia visualisoinnin näkyvästä alueesta. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos puita ei ole olemassa, ja poikkeuksen *InvalidSizeException*, jos ja vain jos puun koko yritetään laittaa liian suureksi tai pieneksi.

```
double getTreeSizeByPixels(int location)
    throws NonexistentTreeException
```

Palauttaa sijainnissa *location* olevan puun koon pikseleinä, jos puu on olemassa. Muutoin metodi heittää poikkeuksen *NonexistentTreeException*.

```
double getTreeSizeByPercentage(int location)
    throws NonexistentTreeException
```

Palauttaa sijainnissa *location* olevan puun koon prosentteina visualisoinnin kokonaisalueesta, jos puu on olemassa. Muutoin metodi heittää poikkeuksen *NonexistentTreeException*.

```
double getTreeSizeByPercentageOfVisibleArea(int location)
    throws NonexistentTreeException
```

Palauttaa sijainnissa *location* olevan puun koon prosentteina visualisoinnin näkyvästä alueesta, jos puu on olemassa. Muutoin metodi heittää poikkeuksen *NonexistentTreeException*.

```
void setSelection(Rectangle2D selection)
    throws IndexOutOfBoundsException
```

Asettaa alueen *selection* valituksi. Poistaa valinnan, jos *selection = null*. Vain yksi alue kerrallaan voi olla valittuna. Metodi heittää poikkeuksen *IndexOutOfBoundsException*, jos ja vain jos *selection* ei sisälly klusterointiin.

```
Rectangle2D getSelection()
```

Palauttaa valitun alueen.

```
void setColoring(GradientColorPalette color)
```

Asettaa lämpökartan väritykseksi paletin *color*.

```
GradientColorPalette getColoring()
```

Palauttaa lämpökartan väripaletin.

```
DataRange getDatasetRowsAtHeatMapRow(int row)
```

Palauttaa tiedon siitä, mitkä data-alueen rivit sisältyvät visualisoinnin riviin *row*.

```
DataRange getDatasetColumnsAtHeatMapColumn(int column)
```

Palauttaa tiedon siitä, mitkä data-alueen sarakkeet sisältyvät visualisoinnin sarakkeeseen *column*.

```
int getHeatMapRowCount()
```

Palauttaa näkyvillä olevan lämpökartan rivien määrän.

```
int getHeatMapColumnCount()
```

Palauttaa näkyvillä olevan lämpökartan sarakkeiden määrän.

```
void setMouseOverHighlight(boolean highlight)
```

Asettaa klusteripuun haarojen korostamisen. Jos *highlight = true*, klusteripuun ne haarat korostetaan, jotka vastaavat hiiren kursorin sijaintia lämpökartalla.

```
boolean getMouseOverHighlight()
```

Palauttaa *true*, jos klusteripuun haarojen korostus on käytössä, muuten *false*.

```
void setTreeHighlight(int location, int index)
    throws NonexistentTreeException,
           IndexOutOfBoundsException
```

Asettaa sijainnissa *location* olevan puun niihin osiin korostuksen, jotka liittyvät lämpökartassa valittuna olevaan riviin tai sarakkeeseen *index*. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos sijainnissa *location* ei ole puuta.

```
int getTreeHighlight(int location)
    throws NonexistentTreeException
```

Palauttaa tiedon puun korostuksesta. Metodi heittää poikkeuksen *NonexistentTreeException*, jos ja vain jos sijainnissa *location* ei ole puuta.

```
void setExpansionState(HCNodeInfo node,
                       boolean expand,
                       boolean recursive)
```

Jos *expand = true*, avaa haaran *node*, muuten sulkee. Jos *recursive = true*, suoritetaan operaatio myös alihaaroille.

```
void draw(Graphics2D g2,
           Rectangle2D area,
           Point2D anchor,
           PlotState parentState,
           PlotRenderingInfo info)
```

Piirtää hierarkkisen puun ja lämpökartan halutun alueen *area* sisälle.

```
HCTreeNodeEntity drawSubTree(Graphics2D g2,
                              HCNodeInfo node,
                              Rectangle2D area)
```

Piirtää rekursiivisesti puun *node* alueelle *area*.

```
void redrawNode(HCTreeNode node)
```

Piirtää yksittäisen solmun *node*.

```
void drawBlock(Graphics2D g2, int x, int y)
```

Piirtää lämpökartan solun  $(x, y)$ .



#### 4.2.6 HCNodeInfo

Luokka toteuttaa puun, joka on rakenteeltaan identtinen Dataset-puun kanssa. Se sisältää visualisoinnissa käytettävän solmuihin liittyvän tiedon. Se on toteutettu, jotta malli-näky-mä-ohjain-arkkitehtuuri voidaan pitää mahdollisimman puhtaana.

Luokan konstruktori on:

```
HCNodeInfo(HCTreeNode node)
```

Asettaa solmun *node*.

Luokka toteuttaa seuraavat julkiset metodit:

```
boolean setNodeOpen(boolean open)
```

Asettaa solmun avoimeksi, jos *open = true*, muuten asettaa solmun suljetuksi.

```
boolean isNodeOpen()
```

Palauttaa *true*, jos tämä solmu on avattuna, muuten *false*.

```
HCTreeNode getNode()
```

Metodi palauttaa tähän solmuun liittyvän datasolmun.

```
void setLeftChild(HCNodeInfo node)
```

Asettaa tämän solmun vasemmaksi lapseksi solmun *node*.

```
HCNodeInfo getLeftChild()
```

Palauttaa tämän solmun vasemman lapsen.

```
void setRightChild(HCNodeInfo node)
```

Asettaa tämän solmun oikeaksi lapseksi solmun *node*.

```
HCNodeInfo getRightChild()
```

Palauttaa tämän solmun oikean lapsen.

```
void setParent(HCNodeInfo node)
```

Asettaa tämän solmun vanhemmaksi solmun *node*.

```
HCNodeInfo getParent()
```

Palauttaa tämän solmun vanhemman.

#### 4.2.7 HCNodeInfoRoot

*extends HCNodeInfo*

Luokka täydentää HCNodeInfo-luokkaa siten, että sen avulla voidaan liittää HCNodeInfo-luokan juurisolmuihin tieto HCPlot-oliosta, johon ne kuuluvat.

Luokan konstruktori on:

```
HCNodeInfoRoot(HCPlot plot)
```

Asettaa *HCPlot* olion *plot*.

Luokka toteuttaa seuraavan julkisen metodin:

```
HCPlot getPlot()
```

Palauttaa *HCPlot*-olion, johon solmu kuuluu.

#### 4.2.8 HCTreeNodeEntity

*extends ChartEntity*

Luokka ylläpitää puun solmun visualisoinnissa tarvittavaa tietoa ja toimii osana eventien käsittelyä.

Luokan konstruktori on:

```
HCTreeNodeEntity(Shape area,
                  String toolTipText,
                  String urlText,
                  Point center,
                  Rectangle subTreeArea,
                  HCNodeInfo node)
```

Luo uuden entiteetin alueelle *area* solmulle *node* pisteeseen *center*. Yhteensä solmu ja sen alipuu varaavat tilan *subTreeArea*. Liittää entiteettiin työkaluvihjeen *toolTipText* ja mahdollisen HTML-imagemap-tekstin *urlText*.

Luokka toteuttaa seuraavat julkiset metodit:

```
HCNodeInfo getHCNodeInfo()
```

Metodi palauttaa tähän solmuun liittyvän puuntilasolmun.

```
Rectangle2D getSubTreeArea()
```

Metodi kertoo tämän solmun ja siitä lähtevän alipuun näytöltä varaaman tilan.

```
Point getCenter()
```

Palauttaa symbolin keskipisteen sijainnin.

#### 4.2.9 HeatMapBlockEntity

*extends ChartEntity*

Luokka ylläpitää lämpökartan solun visualisoinnissa tarvittavaa tietoa ja käsittelee even-  
tejä.

Luokan konstruktori on:

```
HeatMapBlockEntity(Shape area,
                   HCDataset dataset,
                   int x,
                   int y,
                   String toolTipText)
```

Luo uuden entiteetin Java2D-avaruuden alueelle *area* HCDataset-olion *dataset* lämpökartan soluun sarakkeessa *y* ja rivillä *x*. Tähän se liittää työkaluvihjeen *toolTipText* ja mahdollisen HTML-imagemap-tekstin *urlText*.

#### 4.2.10 GradientColorPalette

GradientColorPalette tuottaa tasaisesti liukuvan värisarjan vapaavalintaisille avainarvoille asetettujen värien välille.

Luokan konstruktori on:

```
GradientColorPalette()
```

Luokka toteuttaa seuraavat julkiset metodit:

```
void addKeyColor(double value, int r, int g, int b)
```

Lisää palettiin lukuarvon *value* yhdistettynä väriin  $(r, g, b)$ .

```
void addKeyColor(double value, Color color)
```

Lisää palettiin lukuarvon *value* yhdistettynä väriin *color*.

```
Color getKeyColor(int index)
```

Palauttaa listassa järjestysnumerolla *index* esiintyvän avainvärin. Palauttaa arvon *null*, jos *index* on suurempi tai yhtäsuuri kuin paletin avainvärien määrä tai pienempi kuin 0.

```
Color removeKeyColor(int index)
```

Poistaa ja palauttaa paletista järjestyksellä *index* esiintyvän avainvärin. Palauttaa arvon *null*, jos *index* on suurempi tai yhtäsuuri kuin paletin avainvärien määrä tai pienempi kuin 0. Järjestyksessä myöhemmin esiintyvien avainvärien indeksit vähenevät 1:llä.

```
Color getColor(double value)
```

Palauttaa lukuarvoa *value* vastaavan värin.

```
void setLogarithmic(double value)
    throws IllegalArgumentException
```

Asettaa logaritmisen väriskaalauksen kantaluvulla *value*. Heittää poikkeuksen *IllegalArgumentException*, jos *value*  $\leq 1$ .

```
double getLogarithmic()
```

Palauttaa logaritmisen väriskaalauksen kantaluvun.

```
void setLinear(boolean linear)
```

Asettaa lineaarisen väriskaalauksen, jos *linear* = *true*.

```
boolean isLinear()
```

Palauttaa arvon *true*, jos lineaarinen väriskaalaus on käytössä, muuten arvon *false*.

#### 4.2.11 HCMediator

Luokka on toimii välittäjä CategoryAxis- ja HCPlot-luokkien välillä.

Luokan konstruktori on:

```
HCMediator(HCNodeInfo nodeInfo)
```

Parametri *nodeInfo* määrittelee puun, jonka pohjalta luokan edustamat kategoriat määritellään.

Luokka toteuttaa metodin:

```
List getCategories()
```

Palauttaa listan välittäjäluokan edustamista kategorioista, eli lämpökartan rivin tai sarakkeen nimistä.

#### 4.2.12 HCToolTipGenerator

Luokka on rajapintaluokka, jonka avulla voidaan määrittää visualisoinnin työkaluvihjeet.

Luokka esittelee metodin:

```
String generateToolTip(HCDataSet dataset, int row, int column)
```

Palauttaa rivin *row* ja sarakkeen *column* nimet.

#### 4.2.13 StandardHCToolTipGenerator

*implements HCToolTipGenerator*

Luokan konstruktori on:

```
StandardHCToolTipGenerator()
```

Luokka toteuttaa metodin:

```
String generateToolTip(HCDataSet dataset, int row, int column)
```

Palauttaa rivin *row* ja sarakkeen *column* nimet.

### 4.3 Komponentin toiminta

Tässä luvussa käydään läpi joidenkin monimutkaisten luokkien toimintaa.

#### 4.3.1 HCPlot

Tämä on hierarkkisen puun ja lämpökartan visualisoinnin graafisen osion toteuttava luokka. Karkealla tasolla olion toiminta on seuraava:

1. Oliosta luodaan yksi kopio yhtä visualisointia kohden.
2. Tulostettaessa visualisointi joko näytölle, kirjoittimelle tai tiedostoon JFreeChart kutsuu olion draw()-metodia.
3. Piirtämiseen käytettävän alueen koon muuttuessa JFreeChart kutsuu myös draw()-metodia.
4. Visualisoinnin käyttäjä käyttää luokan metodeja visualisoinnin tilan muokkaamiseen. HCPlot on visualisoinnin keskeinen käyttöliittymäluokka; muut luokat ovat lähinnä sisäiseen toteutukseen tarkoitettuja.
5. Olio ylläpitää HCNodeInfo-puuta, joka pitää yllä tietoa suljetuista ja avatuista klusteripuun haaroista.

**Metodi draw()** on luokan keskeinen rajapinta JFreeChart-kirjaston muita komponentteja varten. Tätä käytetään kaavion piirtämiseen haluttuun Graphics2D-olioon. Sen parametrit ovat

- Graphics2D g2,
- Rectangle2D area,
- Point2D anchor,
- PlotState parentState ja
- PlotRenderingInfo info

Metodin toiminta selviää karkeasti seuraavasta pseudokoodista.

```
treeAreas[] = laske puulle sopiva alue.
namesAreas[] = laske rivien ja sarakkeiden nimille sopiva alue.
heatMapArea[] = laske lämpökartalle sopiva alue.

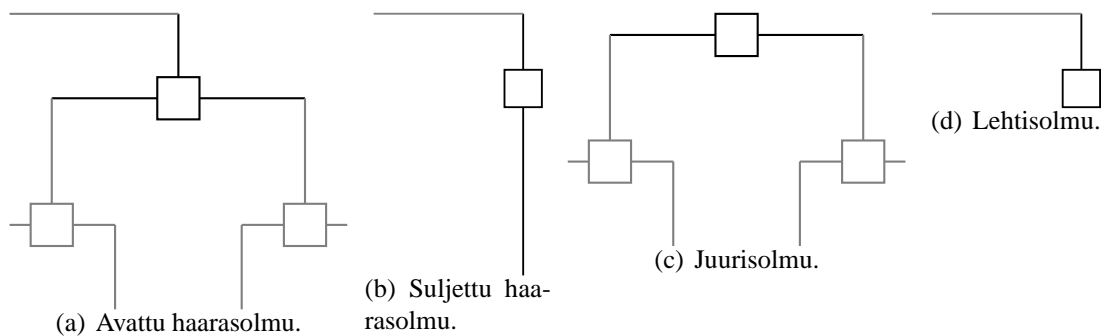
heightOfTree = puun korkeus datasetin haarojen
               height-kenttien summana;
sizeOfNodeSymbol = skaalataan nodea kuvaava laatikko sopivaksi;
widthOfLine = skaalataan puun haaraa kuvaavan viivan
              leveys sopivaksi;

foreach location in top,left {
    if getTreeVisibility (location) {
        drawClusteringTree(
            dataset.getClusteringTree(location),
            treeAreas[location]
        );
    }
    if getNamesVisibility (location) {
        drawNames(
            dataset.getHeatMap().getNames(location),
            treeAreas[location]
        );
    }
}
height = getHeatMapRowCount();
width = getHeatMapColumnCount();
for (x=0; x<width; x++) {
    for (y=0; y<height; y++) {
        drawBlock(x,y);
    }
}
```

Käytetty `drawSubTree()`-metodi toteuttaa klusterointipuun piirtämisen. Luokassa täytyy toteuttaa puiden piirto kaikkiin suuntiin, mutta selvyys vuoksi tässä luvussa puu kuvataan piirrettäväksi ylhäältä alaspäin, jolloin puun leveys on sen lehtisolmujen lukumäärä, jne. Toteutuksessa kaikki tapaukset on kuitenkin otettava huomioon. Metodien toiminta on seuraava:

```
leftEntity = drawSubTree(g2, node.leftChild, area);
leftArea = leftEntity.getSubTreeArea();
rightEntity = drawSubTree(g2, node.rightChild, area-leftArea);
rightArea = rightEntity.getSubTreeArea();
thisArea = redrawNode(node);
entities.add(createEntity());
return entity;
```

**redrawNode()-metodi** toteuttaa yksittäisen solmun piirtämisen. Yksittäisen solmun esitykseen vaikuttaa, onko kyseessä lehtisolmu, juurisolmu vai haarasolmu. Haara- ja juurisolmun tapauksessa vaikuttaa lisäksi, onko kyseinen solmu auki vai suljettu. Kaikkiaan yksittäisen solmun piirtämiseen voi sisältyä itse solmua kuvaava symboli, siitä ylöspäin lähtevä viiva sekä siitä lapsihaaroihin lähtevät vaakaviivat tai alas lämpökarttaan asti johtava viiva. Kuvassa 14 on esitetty erilaiset solmutyypit.



Kuva 14: Erilaiset solmutyypit.

Mikäli solmu on auki, yksittäinen solmu piirretään seuraavasti:

```
// Selvitetään solmun sijainti (x,y),
// korkeus (width) ja leveys (width) pikseleinä.
y = min(leftArea.getY(),rightArea.getY());

leftSymbol = leftEntity.getArea().getBounds2D();
leftLocation = leftSymbol-olion keskipiste;
rightSymbol = rightEntity.getArea().getBounds2D();
rightLocation = rightSymbol-olion keskipiste;
```

```

x = leftLocation.getX()
  + (rightLocation.getX() - leftLocation.getX())/2;
width = rightLocation.getX() - leftLocation.getX();
height = heightOfTreeInPixels * (heightOfNode / heightOfTree);

// Piirretään solmua kuvaava symboli ja siitä lähtevät haarat.
drawSymbolAt(x,y);
drawBranchLine(centerPoint,-width);
drawBranchLine(centerPoint,width);
drawHeightLine(centerPoint,height);

```

Edellä kuvatussa toteutuksessa solmua kuvaava symboli tulee aina sen lapsia kuvaavien symbolien keskipisteeseen. Toinen mahdollisuus olisi piirtää se solmun alipuiden keskipisteeseen. Mikäli solmu on suljettuna, piirretään ainoastaan sitä kuvaava symboli ja tästä viiva alueen alalaitaan. Mikäli puun haarojen korostus on käytössä, pitää viivojen ja symbolin väri valita lämpökartan valinnan mukaan.

**setHorizontalTreeHighlight() ja setVerticalTreeHighlight()** -metodit asettavat näkyvillä olevista puista korostetuksi valitun rivin tai sarakkeen. Vain yksi rivi ja sarake voi olla korostettuna kerralla. Metodin toiminta on ylhäältä alaspäin piirretyn puun osalta seuraava:

```

oldColumn = getHorizontalTreeHighlight(); // mahd. null
newColumn = columnToHighlight; // mahd. null
// Sekä oldColumn että newColumn voivat olla null, mikä
// pitää ottaa huomioon.
oldNodes = niiden nodejen joukko, joihin kuuluu oldColumn;
newNodes = niiden nodejen joukko, joihin kuuluu newColumn;
nodesToUpdate = (newNodes xor oldNodes) +
  nearestParentThatDoesNotBelongTo(newNodes);
foreach node in nodesToUpdate {
  redrawNode(node);
}

```

**setExpansionState()** avaa tai sulkee puun yksittäisen haaran. Sen toiminta on seuraava:

```

node.open = open;
notifyListeners(new PlotChangeEvent(plot));

```

Tämä on kohtalaisen hidas tapa toteuttaa puun haarojen avaaminen ja sulkeminen. Optimointi jätetään toteutusvaiheeseen. Vaihtoehtoinen toteutus on siis olla lähettämättä PlotChangeEvent-eventtiä ja toteuttaa redrawNode()- ja drawBlock()-metodit, jotka piirtävät visualisoinnin muuttuneet osat uudestaan ja muokkaavat jo olemassaolevia HCTreeNodeEntity- ja HeatMapBlockEntity-olioita.



### 4.3.2 HCTreeNodeEntity

Tämän luokan tehtävänä on ylläpitää puun solmun visualisoinnissa tarvittavaa tietoa ja toimia osana eventien käsittelyä.

Eventien käsittely koostuu seuraavista toimista. Luokan oliot asettuvat kuuntelemaan hiiri-eventiä ja

- kutsuvat `plot.setExpansionState(node, !open)`, sekä
- lähettävät eventin eteenpäin rekisteröityneille kuuntelijoille.

Luokan oliot asettuvat kuuntelemaan hiiri-eventiä ja

- kutsuvat `plot.setSelection()`;
- lähettävät eventin eteenpäin rekisteröityneille kuuntelijoille.

## 5 SOM-kartta

Visualisointia varten SOM-kartan soluille on toteutettu tietorakenne, johon on talletettu solun väri, kuvaus ja datavektori. Nämä solut on asetettu matriisiin ja visualisointi toteutetaan käyttäen tätä rakennetta. Matriisiin alkiot piirretään kuusikulmioina ruudulle tietorakenteen ilmoittaman väritiedon mukaan väritettynä. Kuvaus ja datavektori näkyvät joko kuusikulmiossa tai sitä hiirellä osoittamalla ilmestyvässä työkaluvihjeessä. Visualisoinnin toteutuksen luokkakaavio on esitetty kuvassa 15.

### 5.1 Käyttöliittymä

Kuvassa 16 on esimerkki SOM-kartan visualisoinnista perustilassa: kartta koostuu kuusikulmioista, joiden sisällä on tekstinä tieto solun kuvauksesta tai kuvauksista. Hiiren osoittimen vieminen solun päälle ei aiheuta mitään muotoiluja.

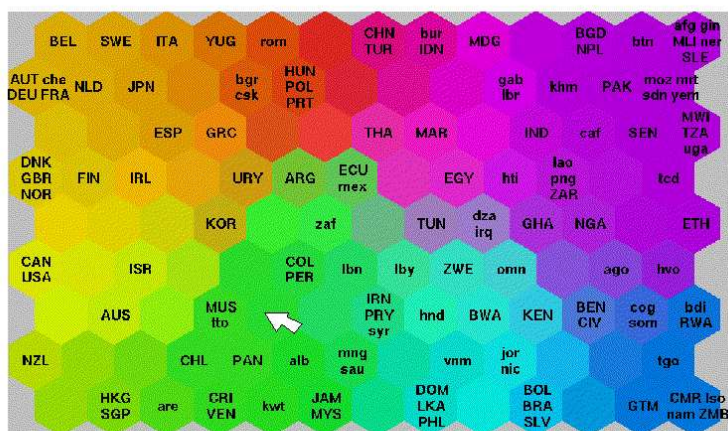
Kun hiiren ykkösnappia painetaan kartan päällä (kuva 17), osoittimen alla oleva solu korostuu reunoistaan. Jonkin toisen solun kohdalla painaminen siirtää valinnan sen kohdalle.

Jos interaktiokäyttäjä haluaa valita useamman solun kerrallaan, hän painaa hiiren ykkösnappia control-nappi näppäimistöltä pohjassa jokaisen sellaisen solun kohdalla, jotka hän haluaa valita. Toinen mahdollinen tapa on control-napin sijasta valita kaksi solua shift-näppäin pohjassa, jolloin ensimmäisen ja toisen väliin jäävät solut — sekä nämä solut itse — valitaan. Näin on tehty kuvassa 18.

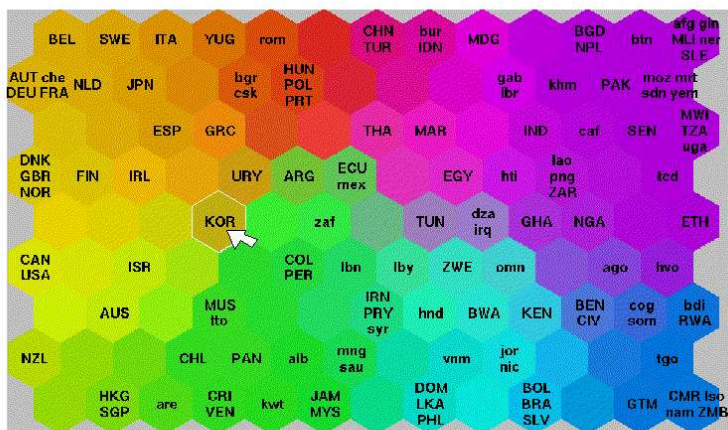
Hiiren kakkosnapin painallus solun päällä valitsee sen ja avaa kontekstivalikon (kuva 19). Properties-valintakohtaa painamalla interaktiokäyttäjä voi alivalinnasta kuvan 20 mukaisella tavalla muuttaa kartan väritystä.

Kuvan 21 solussa on tietoa ollut enemmän kuin soluun on mahtunut. Tällöin näytetään asiaa ilmaiseva symboli. Kun hiiren osoittimen vie symbolilla merkityn solun päälle (kuva 22), työkaluvihje näyttää solun kuvauksen.

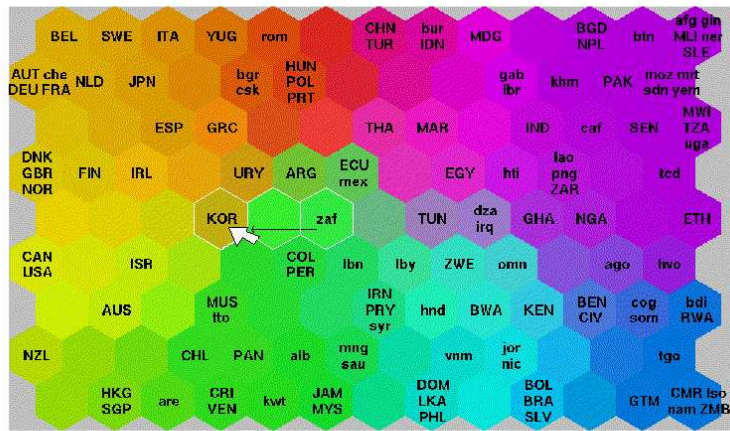




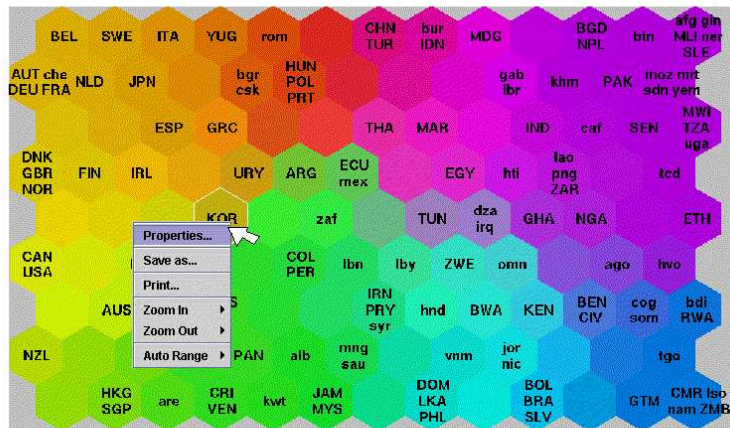
Kuva 16: SOM-kartta perustilassa.



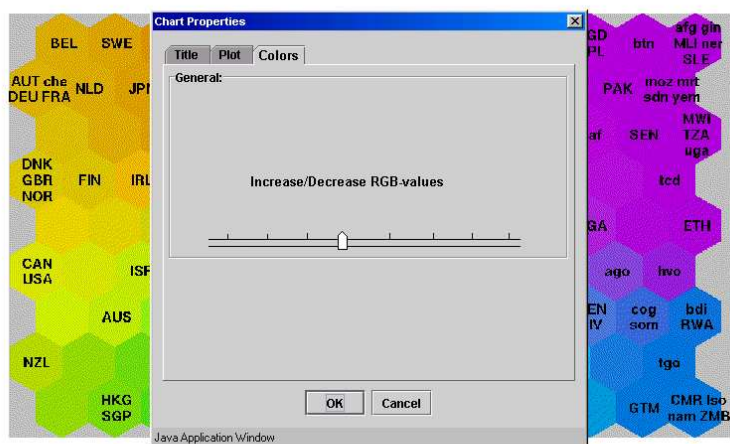
Kuva 17: Yksittäinen kartan solu valittuna.



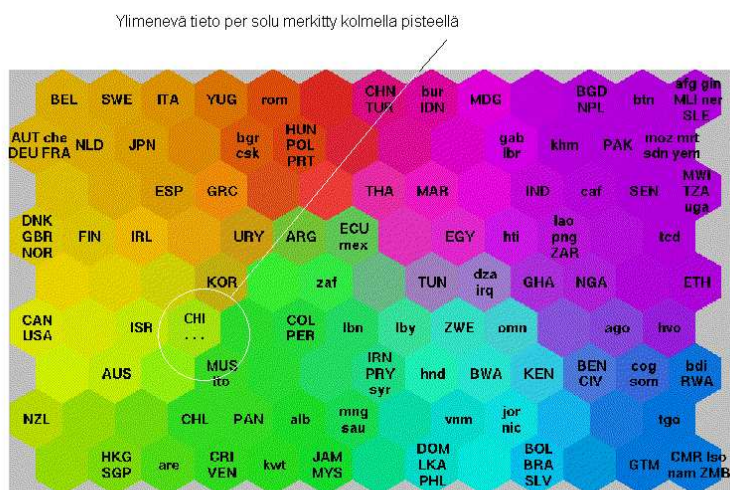
Kuva 18: Useampi solu valittuna shift pohjassa.



Kuva 19: Hiiren kakkospainikkeen painallus ja kontekstivalikko.



Kuva 20: Värin muokkaaminen.



Kuva 21: Informaation ylivuoto yksittäisestä solusta.





## 5.2 Ohjelmointirajapinnat

Tässä kappaleessa esitellään visualisoinnin luokat, niiden sisältämät julkiset metodit, paluarvot ja metodien heittämät poikkeukset niin kuin ne piirretty kuvaan 15.

### 5.2.1 SOMDataItem

Koska yhteen SOM-kartan soluun sisältyy paljon erilaista ja -muotoista dataa, on se helppointa koota omaksi oliokseen. Luokka tallentaa yksittäisen SOM-kartan solun sisältämän datan kenttiinsä. Kenttiä muokataan aksessorimetodeilla.

Luokan konstruktori:

```
SOMDataItem(Color color,
             String descriptions[],
             double[] neuronWeights)
```

Luokka sisältää seuraavat kentät:

Color color

SOM-solun väri *color*.

String[] description

SOM-solun nimi tai nimet taulukossa *description*

double[] neuronWeights

SOM-solun datavektori *neuronWeights*.

Luokan konstruktori on:

```
SOMDataItem(Color color,
             String[] descriptions,
             double[] neuronWeights)
```

Asetetaan dataalkion väri *color*, kuvaukset *descriptions* ja lukuarvot *neuronWeights*.

Luokka sisältää seuraavat metodit:

Color getColor()

Palauttaa SOM-solun väriarvon.



```
String[] getDescription()
```

Palauttaa SOM-solun nimen tai nimet.

```
double[] getNeuronWeights()
```

Palauttaa SOM-solun datavektorin.

### 5.2.2 SOMDataset

*extends AbstractDataset*

SOMDataset sisältää SOM-kartan tietosisällön matriisissa. Matriisin jokainen alkio sisältää SOMDataItem-tietorakenteen, johon on tallennettu kyseisen SOM-kartan alkion väri, datavektori ja nimet. Kartta syötetään matriisiin  $(x, y)$ -parin mukaisesti.

Luokka sisältää seuraavan kentän:

```
SOMDataItem[][] data
```

Matriisi, joka sisältää SOM-kartan solut.

Luokan konstruktori on:

```
SOMDataset(int columns, int rows)
```

Asetetaan uusi matriisi sarakemäärällä *columns* ja rivimäärällä *rows*.

Luokka toteuttaa seuraavat metodit:

```
SOMDataItem getValue(int x, int y)
```

Palauttaa SOM-kartan yksittäisen solun indeksistä *x* ja *y*.

```
int getColumnCount()
```

Palauttaa sarakkeiden lukumäärän.

```
int getRowCount()
```

Palauttaa rivien lukumäärän.

```
void addValue(int x,
              int y,
              Color color,
              String[] description,
              double[] neuronWeights)
    throws IndexOutOfBoundsException
```

Lisää SOM-solun matriisiin kohtaan  $x, y$ , värillä *color*, kuvauksella *description* ja datavektorilla *neuronWeights*. *IndexOutOfBoundsException*-poikkeus heitetään, jos ja vain jos  $(x, y)$  osoittaa matriisin ulkopuolelle.

```
void addValue(int x,
              int y,
              SOMDataItem item)
    throws IndexOutOfBoundsException
```

Lisää SOM-solun *item* matriisiin kohtaan  $x, y$ . *IndexOutOfBoundsException*-poikkeus heitetään, jos ja vain jos  $(x, y)$  osoittaa matriisin ulkopuolelle.

```
List getNeighbors(int x,
                  int y,
                  int maxDistance,
                  boolean includeCenter)
    throws IndexOutOfBoundsException
```

Palauttaa matriisin kohdasta  $x, y$  haettavasta solusta väriarvoltaan etäisyydellä *maxDistance* olevat solut. *IndexOutOfBoundsException*-poikkeus heitetään, jos ja vain jos  $(x, y)$  osoittaa matriisin ulkopuolelle.

```
void changeHueValues(int amount)
```

Muuttaa SOM-solujen väriarvoja arvon *amount mod 360* verran.

```
Iterator iterator()
```

Palauttaa iteraattorin, joka käy läpi kaikki luokan omistamat *SOMDataItem*-oliot.

### 5.2.3 SOMPlot

*extends Plot*

SOMPlot on JFreeChart-kirjaston mukainen tapa koota yhteen data ja sen piirtävä metodi tai metodit. SOMPlot on MVC-arkkitehtuurimallin näkymäkomponentti.

Luokan konstruktori:

```
public SOMPlot(SOMDataset dataset)
```

Luokka sisältää seuraavan kentän:

```
SOMDataset dataset
```

Piirrettävä data.

Luokan konstruktori on:

```
SOMPlot(SOMDataset dataset)
```

Asettaa kentäkseen SOMDataset-olion *dataset*.

Luokka toteuttaa seuraavat metodit:

```
SOMDataset getDataset()
```

Palauttaa piirrettävän datan.

```
void setDataset(SOMDataset dataset)
```

Asettaa SOMPlot-olioon liittyvän SOMDataset-olion *dataset*.

```
void draw(Graphics2D g,
          Rectangle2D area,
          Point2D anchor,
          PlotState parentState,
          PlotRenderingInfo info)
```

Piirtää SOM-kartan annetun alueen *area* sisälle.

#### 5.2.4 SOMToolTipGenerator

Rajapintaluokka työkaluvihjeiden määrittämiseen SOM-kartassa. Luokka tarvitaan, jotta voidaan noudattaa JFreeChart-kirjaston tapaa toteuttaa työkaluvihjeet.

```
String generateToolTip(SOMDataset dataset, int x, int y)
```

Toteutuksen on tarkoitus palauttaa SOMDataset-olion *dataset* sisältämän kaksiulotteisen matriisin  $(x, y)$ -paikalla olevan SOM-solun nimen tai nimet sekä datavektorin arvot.

#### 5.2.5 StandardSOMToolTipGenerator

*implements SOMToolTipGenerator*

Luokan konstruktori on:

```
StandardSOMToolTipGenerator()
```

Luokka toteuttaa metodin:

```
String generateToolTip(SOMDataset dataset,
                      int x,
                      int y)
```

Palauttaa SOMDataset-olion *dataset* sisältämän kaksiulotteisen matriisin *x*, *y* paikalla olevan SOM-solun nimen tai nimet sekä datavektorin arvot.

### 5.2.6 SOMItemEntity

*extends ChartEntity*

Luokka tiettyä SOM-kartan solua koskevien eventien käsittelyyn. Näitä ovat esimerkiksi hiiren napin painaminen kursorin ollessa visualisoinnin osan päällä tai uuden datan syöttö kuvaan. Yhteistä näille toiminnoille on, että visualisoinnin tulee pystyä jakamaan osiin (esim. SOM-kartan yksittäinen solu), joita voidaan käsitellä itsenäisinä entiteetteinä, so. SOMItemEntity-olioina.

Luokan konstruktori on:

```
SOMItemEntity(Shape area,
              SOMDataset dataset,
              int x, int y,
              String toolTipText,
              String urlText)
```

Luo uuden entiteetin Java2D-avaruuden alueelle *area* SOMDataset-olion *dataset* matriisin paikkojen *y* ja *x* SOMDataItem-oliota vastaamaan. Tähän se liittää työkaluvihjeen *toolTipText* ja mahdollisen HTML-imagemap tekstin *urlText*.

Luokka sisältää seuraavat metodit:

```
SOMDataset getDataset()
```

Palauttaa entiteettiin liittyvän SOMDataset-olion.

```
int getX()
```

Palauttaa entiteetin paikan *x* matriisissa, johon se viittaa.

```
int getY()
```

Palauttaa entiteetin paikan *y* matriisissa, johon se viittaa.

### 5.3 Komponentin toiminta

Komponentti toteuttaa JFreeChart-kirjaston mukaisen visualisoinnin. Näissä visualisoinnissa on aina kaksi keskeistä luokkaa: plot ja dataset, joista ensimmäisessä ovat piirtorutiinit ja toisessa visualisointiin liittyvä data. SOM-kartan kohdalla nämä luokat ovat *SOMPlot* ja *SOMDataset*.

*SOMDataItem* on apuluokka, joka sisältää yhden SOM-kartan solun datan omana tietorakenteenaan. *SOMDataset*-luokan matriisi sisältää *SOMDataItem*-olioita, joita *SOMPlot* käyttää piirtäessään visualisointia ruudulle.

## 6 Visualisointikokoelma

Visualisointikokoelma antaa JFreeChart-kirjastolle tuen tulostaa ja tallentaa monta visualisointia eri JPanel-olioista samaan kohteeseen. Tässä käytetään apuna arkkitehtuurikäyttäjän kokoamaa listaa MultiChartPanel-olioista, johon nämä kerätään tulostus/tallennustapahtumaa varten. MultiChartPanel-luokka on laajennus JFreeChart-kirjastossa esiintyvistä ChartPanel-luokasta. Kokoelman toteuksen luokkakaavio on esitetty kuvassa 23.

### 6.1 Ohjelmointirajapinnat

Tässä kappaleessa esitellään komponentin luokat ja niiden sisältämät julkiset metodit, keskeiset kentät sekä JFreeChart-kirjaston luokkiin tehdyt lisäykset.

#### 6.1.1 MultiChartPanel

*extends ChartPanel*

Luokka mahdollistaa useamman MultiChartPanel-olion liittämisen yhteen. Alkuperäiseen luokkaan tarvittavat muutokset tehdään perittyyn luokkaan, jolloin arkkitehtuurikäyttäjän tietää selkeästi, milloin hän on mahdollistamassa useamman visualisoinnin yhdistämisen. Lisäksi ChartPanel-luokan periminen mahdollistaa yhteensopivuuden JFreeChart-kirjaston kanssa.

Luokan konstruktorit ovat:

```
public MultiChartPanel(JFreeChart chart,
                        List panels)

public MultiChartPanel(JFreeChart chart,
                        boolean useBuffer,
                        List panels)

public MultiChartPanel(JFreeChart chart,
                        boolean properties,
                        boolean save,
                        boolean print,
                        boolean zoom,
                        boolean tooltips,
                        List panels)

public MultiChartPanel(JFreeChart chart,
                        int width,
                        int height,
                        int minimumDrawWidth,
```

```

        int minimumDrawHeight,
        int maximumDrawWidth,
        int maximumDrawHeight,
        boolean useBuffer,
        boolean properties,
        boolean save,
        boolean print,
        boolean zoom,
        boolean tooltips,
        List panels)

public MultiChartPanel(CharPanel chartpanel,
        List panels)

```

Konstruktorit ovat pääasiassa vastaavia kuin *ChartPanel*-luokassa, mutta jokaiseen on lisätty *panels*, joka osoittaa listaan, jossa visualisointikokoelmaan kuuluvat paneelit ovat. Tätä listaa käytetään apuna tulostuksessa ja tallennuksessa. Viimeinen konstruktori mahdollistaa olemassa olevan *ChartPanel*-olion muuttamisen *MultiChartPanel*-olioksi.

Luokka sisältää seuraavan kentän:

```
List panels
```

Viittaa listaan, jossa kaikki tallennuksessa ja tulostuksessa käytettävät *MultiChartPanel*-oliot ovat.

Luokka sisältää seuraavat metodit:

```
void print(Graphics g, PageFormat pf, int pageIndex)
```

Tulostaa listan sisältämien *MultiChartPanel*-olioiden visualisoinnit samaan kohteeseen.

```
void doSaveAs()
```

Tallentaa listan sisältämien *MultiChartPanel*-olioiden visualisoinnit samaan kohteeseen.

### 6.1.2 MultiChartUtilities

*MultiChartUtilities* on kirjastoluokka, joka tarjoaa julkiset metodit useiden *JFreeChart*-olioiden tulostuksen samaan kohteeseen.

Luokka sisältää seuraavat metodit:

```

void saveChartsAsPNG(File file,
        List charts,
        int width,
        int height)
    throws IOException

```

Tallentaa listan *charts* visualisoinnit samaan PNG-formaatissa olevaan tiedostoon *file*. Kuvan leveys määräytyy parametrilla *width* ja korkeus parametrilla *height*.

```
void writeChartsAsPNG(OutputStream out,
                     List charts,
                     int width,
                     int height)
    throws IOException
```

Kirjoittaa listan *charts* visualisoinnit samaan kohteeseen (OutputStream *out*) leveydellä *width* ja korkeudella *height*.

```
void saveChartsAsJPEG(File file,
                     List charts,
                     int width,
                     int height)
    throws IOException
```

Tallentaa listan *charts* visualisoinnit samaan JPEG-formaatissa olevaan tiedostoon *file* leveydellä *width* ja korkeudella *height*.

```
void writeChartsAsJPEG(OutputStream out,
                     List charts,
                     int width,
                     int height)
    throws IOException
```

Kirjoittaa listan *charts* visualisoinnit samaan kohteeseen (OutputStream *out*) leveydellä *width* ja korkeudella *height*.

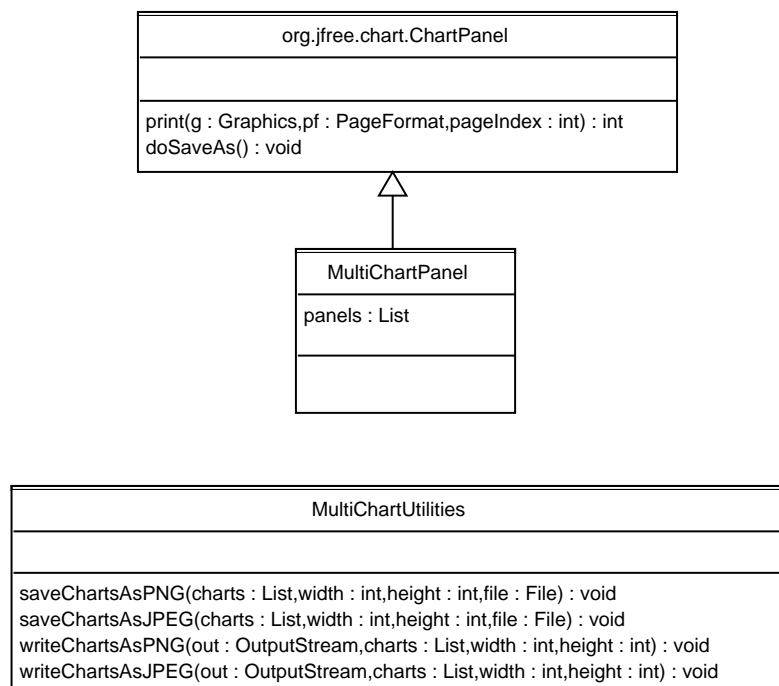
## 6.2 Komponentin toiminta

Kun käyttäjä haluaa tulostaa tai tallentaa monen *ChartPanel*-olion sisällä olevat visualisoinnit, käyttäjä luo *ChartPanel*-olion sijasta *MultiChartPanel*-olion tai muuntaa olemassa olevan *ChartPanel*-olion *MultiChartPanel*-olioksi.

*MultiChartPanel*-olio sisältää viitteen listaan kaikista muista samaan aikaan käsiteltävistä *MultiChartPanel*-olioista. Tätä listaa pitää yllä arkkitehtuurikäyttäjä. Listaa apuna käyttäen toteutetaan kaikkien *MultiChartPanel*-ilmentymien sisältämien visualisointien tulostaminen ja tallentaminen samaan kohteeseen siten, että aina kun yhdestä listaan kuuluvasta visualisoinnista tulostetaan tai tallennetaan, käydään läpi koko lista ja tulostetaan tai tallennetaan kaikki siitä löytyvät visualisoinnit. Nämä asemoidaan seuraavasti: kaksi vierekkäin ja  $n$  kappaletta allekain.



MultiChartPanel-oliassa ChartPanel-luokan print() ja doSaveAs() korvataan uusilla metodeilla, jotka osaavat käyttää hyväkseen listan tietoja. doSaveAs() käyttää *MultiChartUtilities*-luokan metodeita apunaan tallettaessaan useita visualisointeja samaan kohteeseen.



Kuva 23: Visualisointikokoelma

## **7 Kolmiulotteinen hajontakuvio**

Tämä visualisointi suunnitellaan ja toteutetaan myöhemmin, mikäli projektin aikataulu antaa myöten.