Kuntokirjuri

Ohjelmistokuvaus

Miika Alonen

Jarkko Laine

Jesse Honkanen

Veli-Matti Huovinen

Jani Jäntti

Versio 1.0

9.5.2008

Jakelu:

Asiakas - Jukka Rantala

Ohjaaja – Erkki Pesonen

Opponoiva ryhmä 1

Kuopion yliopisto

tietojenkäsittelytieteen laitos

Dokumentin versiohistoria:

Versio	Pvm	Tekijä	Muutos
0.1	19.4.08	MA	Yleistä ohjelmistokuvauksesta
0.2	20.4.08	JH	Lisää yleistä tietoa
0.3	21.4.08	JH	Hieman muokkausta
0.4	22.4.08	MA	Tarkempaa tietoa ohjelman rakenteesta
0.5	23.4.08	MA	Lisää tarkempaa tietoa
1.0	9.5.08	MA	Kirjoitusasun korjauksia

Tekijöiden lyhenteet:

MA	Miika Alonen
JJ	Jani Jäntti
JH	Jesse Honkanen
VH	Veli-Matti Huovinen
JL	Jarkko Laine

SISÄLLYSLUETTELO

1 JOHDANTO

1.1 Tarkoitus ja kattavuus

Tässä dokumentissa on kuvattu **Kuntokirjuri**-ohjelman toteutus. Dokumentti on tarkoitettu mahdolliselle jatkokehittäjälle tai ylläpitäjälle. Tämän dokumentin lisäksi projektin koodissa on **Javadoc**-kommentointi [10], joten tarkemmat metodikuvaukset löytyvät koodin mukana tulevista Javadoc-sivuista. Myös muista projektin aikana syntyneistä dokumenteista löytyy hyödyllistä tietoa järjestelmän toiminnasta. Muut dokumentit ovat saatavissa samassa paikassa kuin tämä dokumentti, eli projektin verkkosivuilta [1]. Näistä dokumenteista löytyy myös järjestelmän yleiskuvaus ja mahdollisia jatkokehitysajatuksia. Tässä dokumentissa on lyhyitä arvioita tehdyistä ratkaisuista, mutta ei kuvauksia vaihtoehtoisista ja hylätyistä ratkaisuvaihtoehdoista.

1.2 Dokumentin rakenne

Luvussa 2 on listattu käytetyt välineet ja komponentit

Luvussa 3 on annettu ohjelmarakenteen yleiskuvaus

Luvussa 4 on kuvattu ohjelmiston arkkitehtuuri

Luvussa 5 on ohjeita ylläpitoon ja kehittämiseen

2 TOTEUTUSYMPÄRISTÖ

Tässä luvussa on kuvattu ohjelman kehittämiseen käytetyt välineet ja tarvittava ympäristö.

2.1 Käytetyt kehitysvälineet

Netbeans 6.0.1 [6] – Monipuolinen avoimen lähdekoodin sovelluskehitysympäristö

JDK 1.6 [8]– Sunin virallinen java-kehityspaketti

iReport 0.9.1 nb plugin [4] – Avoimen lähdekoodin raporttipohjien tekotyökalu, josta saatavilla sekä Netbeans ympäristöön integroituva paketti että erillinen ohjelma

2.2 Käytetyt komponenttikirjastot

Ohjelma käyttää useita valmiita komponenttikirjastoja eri avoimen lähdekoodin projekteista, kuten SwingX, JasperReports ja JDateChooser. Myös näillä projekteilla on riippuvuuksia lukuisiin muihin avoimen lähdekoodin projekteihin. Alla on listattu toteutuksessa käytetyt komponenttikirjastot. Jokainen paketti on lisensoitu avoimen lähdekoodin lisenssillä (Lisenssi suluissa).

Swing Application Framework appframework 1.0.3 - (GPL)

Swing Application Framework swing-worker 1.1 - (GPL)

JasperReports 2.0.3 [3]– Raportointityökalut (GPL)

DateChooser 1.1 – Kalenterityökalu (Sun Public Licence)

SwingX 27-1-2008 – Komponentteja käyttöliittymän tekoon (LGPL)

Substance Lite 4.3 – Ulkoasukomponentit (BSD)

Derby 10.3.2.1 [7] – Käytetty JDBC-tietokantamoottori (Apache)

2.3 Yhteensopivuus

Käytettyjen välineiden yhteensopivuutta aiempien versioiden kanssa ei ole kokeiltu. Toteutuksessa käytettiin tuoreimpia osia tarkoituksella, sillä taaksepäin yhteensopivuutta ei pidetty erityisen tärkeänä. Kehitysvälineet ovat kaikki java-pohjaisia ja saatavilla myös muille alustoille kuin Windowsille.

3 TOTEUTUKSEN YLEISKUVAUS

Kuntokirjuri-ohjelma on "**Swing Application Framework**"-arkkitehtuurimallin mukainen työpöytäsovellus. Ohjelman perusidea on toimia kalenteripohjaisena terveystietojen seurantajärjestelmänä, johon voidaan helposti lisätä ja poistaa merkintöjä, merkintätyyppejä ja niihin liittyviä ominaisuuksia.

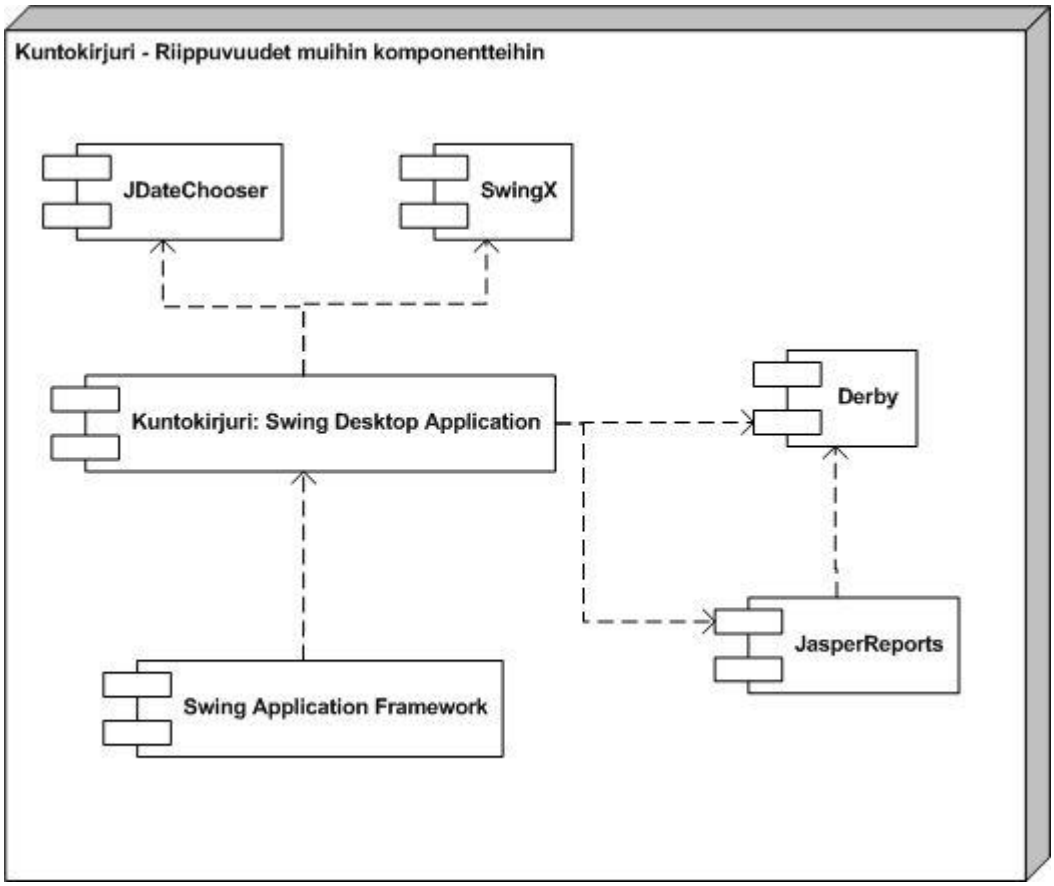
Talletettavat tiedot säilytetään ohjelmiston sulautettuun **Derby**-tietokantaan. Merkintöjen tallentamiseen ja hakemiseen tietokannasta käytetään apuna *Markings*-luokkaa, josta luodaan merkintäolioita. Näihin olioihin yksilöidään tietokannasta kaikki yhteen merkintään liittyvät arvot. Markings-luokka toimii siis rajapintana ja tietojen yhdistäjänä relaatiotietokannalle, jossa tiedot on hajautettu tarvittavan joustavuuden, kuten merkintätyyppien ja ominaisuuksien "lisäys ja poisto"-toimintojen saavuttamiseksi.

Database-luokassa määritellään tietokannan rakenne, salausmenetelmä ja tarvittavat metodit tietojen lisäämiseen ja poistoon. Valmiit Markings-oliot tuodaan tietokannasta ohjelmassa käytettyjen komponenttien "malli"-luokkiin jonka avulla voidaan helposti päivittää komponenttien tietoja. Tällaisia malliluokkia on esimerkiksi *MarkingTableModel* ja *MarkingListModel*. Käyttöliittymän tapahtumakuuntelijoilla päivitetään "malli"-luokkien tietoja, tuomalla niihin Markings-oliotaulukoita, jolloin myös niihin liittyvät käyttöliittymän komponentit päivittyvät automaattisesti.

Ohjelmassa käytetyt raportit luodaan suoraan tietokannasta käyttäen **JasperReport**-komponenttikirjastoa. Ohjelmassa käytetyt valmiit raporttipohjat ohjelmaan on luotu Netbeans:in **iReport**-lisäkomponentilla.

4 ARKKITEHTUURI

Tässä luvussa on kuvattu ohjelmiston arkkitehtuuri, esittelemällä ohjelmistossa käytetyt luokat, tietokanta käyttöliittymä ja raporttitoiminto. Suurella abstraktiotasolla ohjelmiston arkkitehtuuri koostuu alla olevista komponenteista.



Kuva 1: Komponenttikaavio UML-notaatio

4.1 Luokkakuvaukset

Tässä osassa on kuvattu ohjelman luokkien käyttötarkoitus. Tarkemmat kuvaukset luokista löytyy lähdekoodien mukana tulevista **Javadoc**-dokumentaatiosta. Liitteissä on tiivistetty luokkakaavio joka kuvaa ohjelmassa käytetyt luokat, ja niiden tärkeimmät attribuutit ja operaatiot. **Liite 1.**

Kaikki projektissa kirjoitetut luokat on määritelty kuulumaan osaksi ”*kuntokirjurievolution*” -pakettia. Itse luotujen luokkien lisäksi ohjelmassa käytetään lukuisia kirjastoista löytyviä luokkia. Kannattaa ensin tarkistaa JDK:n api [11].

Merkittävät luokat

- KKApplication

  Tämä luokka on Swing Application Framework – mukainen käyttöliittymäluokan yliluokka, joka näyttää kirjautumisikkunan, ja käynnistää ohjelman kun sisäänkirjaus onnistuu.

- KKView

  Tämä luokka on Swing Desktop Application – mukainen käyttöliittymäluokka, johon liittyy olennaisesti KKView.form – tiedosto, jonka Netbeans 6.0 generoi automaattisesti. Ilman form tiedostoa, ohjelmaa ei voida kehittää Netbeans-kehitysympäristössä. Käyttöliittymäluokka sisältää pääosan ohjelmassa käytetyistä komponenteista, mutta on riippuvainen myös muista luokista.

- Database

  Tämä luokka on luo sulautetun tietokannan käyttäen Derby-tietokantamoottoria. Tarkempi kuvaus tietokannasta kohdassa 4.4.

- Markings

  Tästä luokasta luodaan olioita, jotka voi sisältää yhden merkinnän kaikki tiedot. Markings-olioita käytetään tiedon siirtämiseen tietokannan ja ohjelman välillä.

Lomakeluokat

- kuntokirjurievolutionAboutBox

  Tämä luokka on käyttöliittymän AboutBox. Joka antaa käyttäjälle yleistä tietoa ohjelman versiosta ja tekijöistä.

- LogginFrame

  Tämä luokka on ohjelman sisäänkirjautumisikkuna, jonka avulla käyttäjä voi syöttää ohjelmalle oman profiilinsa ja salasanan. Kirjautumisikkuna yrittää avata tietokannan käyttäjältä saamallaan tunnuksella ja salasanalla, ja taustalla kuunteleva käyttöliittymäluokan yliluokka avaa ohjelman kirjautumisen onnistuessa.

- NewProfileDialog

  Tämä luokka on profiilinluonti-ikkuna josta käyttäjä voi luoda itselleen uuden profiilin. Profiilia luotaessa kutsutaan tietokantaluokan metodia joka rakentaa uudelle käyttäjälle oman tietokannan.

- NewPasswordDialog

  Tämä luokka on salasananvaihtoikkuna jolla voidaan vaihtaa salasanan profiiliinsa. Ikkuna kutsuu tietokantaluokan metodia salasanan vaihtamiseen.

- NewReportDialog

  Tämä luokka on raportinluonti-ikkuna, jolla luodaan uusi raportti. Raportti luodaan käyttäen JasperReport-komponentteja ja raportti vaatii iReport –ohjelmalla tehdyt .jasper raporttipohjat.

Muut luokat

- BusyBox

  Tämä luokka on yksinkertainen, “Odota hetki”-ikkuna, joka käynnistetään silloin, kun ohjelma suorittaa aikaavieviä toimenpiteitä.

- MarkingListModel

  Tämä luokka on käyttöliittymän JList-komponentin käyttämä malliluokka, jonka avulla voidaan päivittää tiedot käyttöliittymään näkyville. Tätä mallia voidaan myös kutsua päivittämään MarkingTableModel -malliluokka, jolloin haluttu merkintä näkyy JTable-komponentissa.

- MarkingTableModel

  Tämä luokka on käyttöliittymän JTable-komponentin käyttämä malliluokka, jonka avulla voidaan näyttää yksi Marking-olio halutussa taulussa. Päivittämällä malliluokan sisältämä Marking-olio, päivittyy sen tiedot siihen liittyvään JTable-komponenttiin automaattisesti.

- MultiLineColumn

  Tämä luokka mahdollistaa JTable-komponentin sarakkeen muokkaamisen automaattisesti siten, että jos yhteen kenttään kirjoitetun tekstin määrä ylittää kentän pituuden, muokataan rivin korkeutta, ja asetetaan teksti näkymään useammalle riville.

- DiaryListRenderer

  Tämä luokka mahdollistaa sen, että JList-komponentin objekteja voidaan kuunnella ja tarvittaessa muokata komponentin ulkoasua sekä sen solujen sisältämää tekstiä.

- TextVerifier

  Tällä luokalla voidaan tarkistaa että käyttäjä ei syötä virheellisiä arvoja tekstikenttään.

- NoSqlVerifier

  Tällä luokalla voidaan estää että käyttäjä ei voi syöttää SQL-komentoja tekstikenttään.

- NumberVerifier

  Tällä luokalla voidaan tarkistaa että käyttäjän antama numeraalinen syöte on annetulta väliltä x-y.

- BodyMassIndex

  Tämä luokka sisältää metodit painoindeksin laskemiseen ja arvoihin liittyvään luokitukseen.

4.2 Käyttöliittymä

Käyttöliittymä on tehty Swing Desktop Application ja Swing Application Framework – mukaisesti, Netbeans 6.0 kehitysympäristössä. Käyttöliittymän toteutuksessa käytettiin pääosin Javan **Swing** ja **SwingX** -komponentteja. Asettelut tehtiin Netbeansin suunnittelutyökaluilla. Käyttöliittymän toteutuksessa käytettiin monia ulkopuolisia komponentteja. Joidenkin käyttöliittymän osien kanssa jouduttiin kikkailemaan halutun toiminnan aikaansaamiseksi. Käyttöliittymän tapahtumankäsittelijät

käyttävät hyödykseen *Database*-luokan tarjoamia palveluita markings-luokan ja komponenttien “malli”-luokkien kautta.

4.3 Raportit

Raportit toteutettiin *JasperSoftin*[2] ylläpitämillä avoimen lähdekoodin työkaluilla **JasperReports**[3] ja **iReport**[4]. Raporttipohjat tehtiin iReportin Netbeans-pluginilla. XML-muotoiset raporttipohjat löytyvät *.jrxml* -tiedostoista ja niistä käännetyt tiedostot *.jasper* -tiedostoissa. Pohjia voi muokata haluamakseen. Yleisohjeet työkalujen käyttöön löytyy läheteissä olevista linkeistä [4] ja [5].

4.4 Tietokanta

Tässä osassa on tarkempi kuvaus tietokannasta ja sen käsittelyyn käytetyistä komennoista.

4.4.1 Yleistä

Lyhyt kuvaus tietokannassa käytetyistä tekniikoista. Lisätietoja **Derby**-tietokannan käytöstä löytyy esimerkiksi Apachen Derby-sivuilta [7].

Tässä luvussa olevat komennot on kopioitu *Database*-luokan metodeista. Tarkoitus on antaa yleiskuva käytetyistä komennoista. Tarkempia tietoja Database-luokan metodeista löytyy *Javadoc*eista ja koodia tarkastelemalla. Kaikki tietokantakomennot tehdään *Statement*-olion kautta käyttäen *executeXXX([SQL-komento])* -metodeja. Lauseissa olevat tunnukset ovat parametrien tunnuksia. Komentoihin liittyviä parametrejä on neljää tyyppiä: merkkijonoja, kokonaislukuja, *Markings*-olioita ja *Timestamp*-olioita. Hakasulut tarkoittavat, että koodissa on ehto, jolla suluissa oleva osa voi jäädä väliin.

Tietokannan viite-eheyksien tarkistuksiin ei käytetty omaa koodia, vaan määriteltiin vierasavaimia sisältävät taulut ”*cascade on delete”*-tyyppisiksi. Tietokannan ER-kaavio liitteessä 1.

4.4.2 Tietokannan käsittelyyn käytetyt komennot

Uuden tietokannan luonti

Tietokannan luontivaiheessa määritellään käyttäjätunnus, salasana ja uuden profiilin perusasetukset. Lyhyesti tietokannan luonti tapahtuu olemassa olevaa Derby-yhteyttä käyttäen komennolla, jossa määritellään uusi tietokanta salatuksi, ja käytetään salausavaimena annettua käyttäjätunnusta ja salasanaa.

DriverManager.getConnection(jdbc:derby+usrname;create=true
;dataEncryption=true
;bootPassword=” + usrname +”zXcFrH64Jmd4&¤hnkFvRTygS +pass,usrprops)

Profiilin asetusten haku

values SYSCS_UTIL.SYSCS_GET_DATABASE_PROPERTY('[asetuksen nimi]')

Profiilin asetusten tallennus

Metodi saa parametrikseen Properties-tyyppisen olion, jossa olevat arvot se tallentaa tietokantaan.

call SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY('entry.getKey()','entry.getValue()')

Profiilin salasanan vaihto

CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY('derby.user.usrname','newpw')

Profiilin varmuuskopionti

CALL SYSCS_UTIL.SYSCS_BACKUP_DATABASE(?)

Varmuuskopion tuonti

jdbc:derby:databasename;restoreFrom=backupLocation

4.4.3 Tietokannan taulujen luontikomennot

Types-taulu:

CREATE TABLE Types (

```
TypeName VARCHAR(100) NOT NULL,
TypeInfo VARCHAR(1000) DEFAULT 'EI LISÄTIETOJA',
CalendarID INTEGER NOT NULL,
CONSTRAINT Types_PK PRIMARY KEY (TypeName,CalendarID)
)
```

Attributes-taulu:

```
CREATE TABLE Attributes (
AttribName VARCHAR(100) NOT NULL,
AttribInfo VARCHAR(1000) DEFAULT 'EI LISÄTIETOJA',
CONSTRAINT Attributes_PK PRIMARY KEY (AttribName)
)
```

TypeCollection-taulu:

```
CREATE TABLE TypeCollections (
AttributePlace INTEGER NOT NULL,
Attributes_AttribName VARCHAR(100),
Types_TypeName VARCHAR(100) NOT NULL,
Types_CalendarID INTEGER NOT NULL,
CONSTRAINT TypeCollections_PK
PRIMARY KEY (AttributePlace,Types_TypeName,Attributes_AttribName),
CONSTRAINT TypeCollections_Attributes_FK
FOREIGN KEY (Attributes_AttribName)
REFERENCES Attributes (AttribName) ON DELETE CASCADE,
CONSTRAINT TypeCollections_Types_FK
FOREIGN KEY (Types_TypeName,Types_CalendarID)
REFERENCES Types (TypeName,CalendarID)
ON DELETE CASCADE
)
```

Markings-taulu:

```
CREATE TABLE Markings (
        Time TIMESTAMP NOT NULL,
        TextValue VARCHAR(1000) DEFAULT NULL,
        TargetBoolean VARCHAR(5) DEFAULT 'false',
        Value DOUBLE DEFAULT NULL,
        Types_TypeName VARCHAR(100) NOT NULL,
        Types_CalendarID INTEGER NOT NULL,
        TypeCollections_AttributePlace INTEGER NOT NULL,
        Attributes_AttribName VARCHAR(100) NOT NULL,
        CONSTRAINT Markings_PK
    PRIMARY KEY  (Time, TargetBoolean, TypeCollections_AttributePlace, Types_TypeName),
        CONSTRAINT Markings_TypeCollections_FK
        FOREIGN KEY (TypeCollections_AttributePlace, Types_TypeName, Attributes_AttribName)
        REFERENCES TypeCollections (AttributePlace, Types_TypeName, Attributes_AttribName)
        ON DELETE CASCADE,
        CONSTRAINT Markings_Types_FK
        FOREIGN KEY (Types_Typename, Types_CalendarID)
        REFERENCES Types (TypeName, CalendarID)
        ON DELETE CASCADE
        )
```

4.4.4 Tyyppien ja attribuuttien käsittely

Tyyppien listaus

```
SELECT TypeName FROM Types [WHERE CalendarID=calendarID] ORDER BY TypeName
```

Attribuuttien listaus

SELECT Attributes_AttribName FROM TypeCollections WHERE Types_TypeName='typeName' ORDER BY AttributePlace TAI, jos typeName == null SELECT AttribName FROM Attributes

Tyypin lisäys

INSERT INTO Types values ('newTypeName','newTypeInfo',calendarID)

INSERT INTO TypeCollections values((i+1), 'attribNames[i]', 'newTypeName', calendarID)

Tyypin poisto

DELETE FROM Types WHERE Types.TypeName='typeName'

Tyypin infot

SELECT TypeInfo FROM Types WHERE TypeName='typename'

Attribuutin lisäys

INSERT INTO Attributes values('aName','aInfo')

Attribuutin poisto

DELETE FROM Attributes WHERE Attributes.AttribName='aName'

Attribuutin infot

SELECT AttribInfo FROM Attributes WHERE AttribName='attribname'

4.4.5 Merkintöjen käsittely

Merkinnän lisäys silmukalla Markings-oliosta

INSERT INTO Markings (Time, TextValue, TargetBoolean, Value, Types_TypeName, TypeCollections_AttributePlace, Types_CalendarID, Attributes_AttribName) values ('newMarking.getTimestamp().toString()', 'newMarking.getAttribTextValue(i)','target.toString()', newMarking.getAttribvalue(i), 'newMarking.getTypeName()', (i+1), +calendarID, 'newMarking.getAttribname(i)')

Merkinnän poisto

DELETE FROM Markings WHERE Markings.Time = 'delMarking.getTimestamp().toString()' AND Markings.TargetBoolean = '((Boolean)delMarking.getTargetValue())).toString()' AND Markings.Types_TypeName='delMarking.getTypeName()'

Merkintöjen listaus

SELECT Time, TextValue, TargetBoolean, Value, Types_TypeName, TypeCollections_AttributePlace, Types_CalendarID FROM Markings WHERE Time BETWEEN 'start.toString()' AND 'end.toString()' [AND Types_CalendarID=calendarID] [AND Types_TypeName='tName'] ORDER BY Time, Types_TypeName, TargetBoolean, TypeCollections_AttributePlace

5 YLLÄPITO-OHJEITA

Tässä kappaleessa on kuvattu ylläpito ja kehitysohjeita mahdollisille jatkokehittäjille.

5.1 Projektin dokumentaatio

Projektin julkaistu dokumentaatio löytyy projektin nettisivuilta [1]. Muista dokumenteista voi löytyä sellaista tietoa, mitä tästä on jätetty pois. Projektin lähdekoodi on myös ladattavissa projektin **SourceForge**-sivuilta. Luokkien *Javadoc*-dokumentaatio löytyy käännettynä lähdekoodin mukana.

5.2 Kehitysympäristö

Kehitysympäristöksi suosittelemme **Netbeansia**[6] ja JDK:n versiota **1.6** tai tuoreempaa. Projektin voi avata helposti, kun lähdekoodipaketti on ensin purettu. Lähdekoodit voivat olla myös saatavissa *CVS*-versionhallintaa käyttäen, jos sitä palvelua päätetään projektin lopussa käyttää. SourceForge ohjeistaa versionhallinnan käytössä.

Raporttipohjien laadintaohjelma **iReport** [4] on saatavissa lisäosana Netbeans-ympäristöön. Vaihtoehdoisesti voidaan ladata versio, joka ei tarvitse Netbeansia toimiakseen.

Huom! Projektia avatessa saattaa tulla virheilmoitus löytymättömistä paketeista. Paketit ovat purkukansion libraries -hakemistossa. Virheen saa korjattua painamalla oikealla näppäimellä tuotua projektia ohjeiden mukaan. Kun kaikki paketit ovat löytyneet pitäisi ohjelman olla toimintavalmis.

Ennenkuin aloittaa käyttöliittymän ulkoasun kohennuksen, kannattaa paneelien layout vaihtaa free-layot muotoon, koska käytetty GridBagLayout estää komponenttien vapaan siirtelyn.

5.3 Projektikansion sisältö.

build/

Sisältää käännetyt java-luokat (*.class) ja resurssitiedostot, eli kuvat ym.

libraries/

Sisältää ohjelmassa tarvittavat kolmannen osapuolen paketit. Kuvaus paketeista luvussa 2.2.

nbproject/

Sisältää Netbeansin projektiasetuksia.

Profiles/

Syntyy vasta, kun ensimmäinen profiili luodaan. Jokainen profiili omassa kansiossaan. Kaikki profiiliin liittyvät tiedot on samassa kansiossa. Kaikki tiedostot on Derby-tietokannan luomia.

Raports/

Sisältää XML-muotoiset raporttipohjat (*.jrxml) ja käännetyt raporttipohjat (*.jasper).

src/

Sisältää projektin lähdekoodit (*.java) ja lomakepohjat (*.form). Sisältää myös kuvakkeita.

build.xml

Muokattava tiedosto, jos haluaa tehdä omanlaisensa käännöksen.

manifest.mf

Tarvitaan paketin paketointiin ja ajoon.

Jaettu paketti on *.jar* -tyyppinen tiedosto, jossa on sisällä käännetyt luokkatiedostot ja muut tarvitut tiedostot ja kirjastot.

5.4 Testaus

Netbeansiin on integroitu monia testausta helpottavia toimintoja, kuten **JUnit** [9]. Projektin aikana näitä työkaluja ei juurikaan käytetty, joten regressiotestaus ei suoraan onnistu. Projektin aikana tehdystä testauksesta löytyy tietoa *Testaussuunnitelmasta* ja *Testausraportista*. Testaus tulee jäämään pahasti keskeneräiseksi projektin päättyessä, joten syytä testaukseen on.

LÄHTEET

[1]Kuntokirjurin kotisivu, http://kuntokirjuri.sourceforge.net/

[2] JasperSoftin kotisivu, http://www.jaspersoft.com/JasperSoft_Products.html

[3] JasperReports, http://jasperforge.org/jaspersoft/opensource/business_intelligence/jasperreports/

[4] Ireport, http://www.jasperforge.org/jaspersoft/opensource/business_intelligence/ireport/page.php?name=iReportNB

[5] Jasperreportsin käyttöohje http://jasperforge.org/jaspersoft/opensource/business_intelligence/jasperreports/tutorial.html

[6] Netbeansin kotisivu, http://www.netbeans.org/

[7] Derbyn kotisivu, http://db.apache.org/derby/

[8] JDK lataussivu, http://java.sun.com/javase/downloads/index.jsp

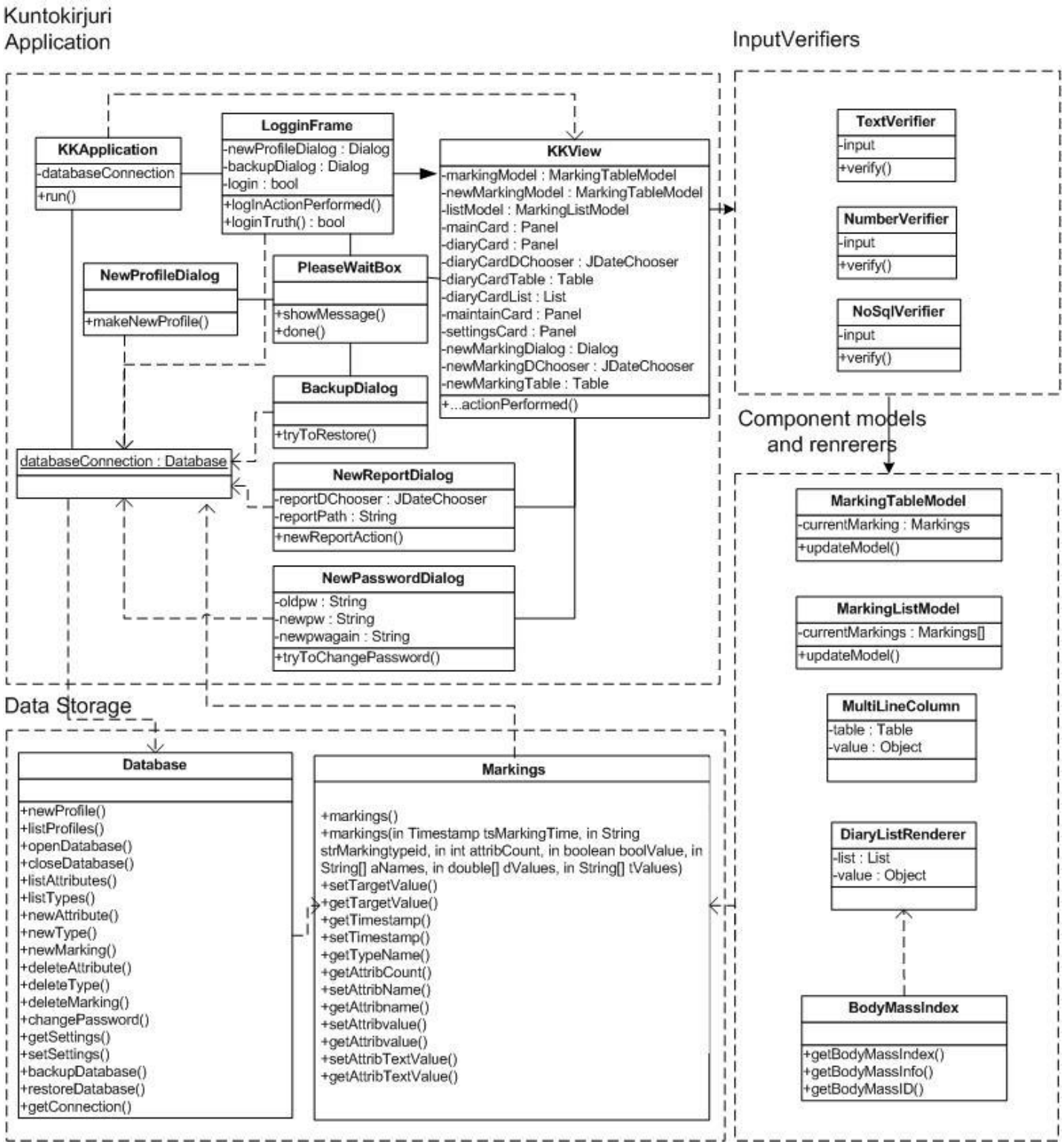
[9] JUnit, http://www.junit.org/

[10] Javadocista tietoa, http://java.sun.com/j2se/javadoc/faq/index.html

[11] Javan API, http://java.sun.com/javase/6/docs/api/

LIITTEET:

Liite 1 – Tiivistetty luokka-kaavio (UML-luokkakaavio, vain tärkeimmät attribuutit ja metodit)



Liite 2 – ER-Kaavio