Ylläpitodokumentti

Karstula

Helsinki 3.5.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti ($9 + 1$ op)

Projektiryhmä

Juha-Pekka Juutilainen
Hannu Kukko
Antto Mäkinen
Antti Rajasärkkä
Ari Raunio
Mika Tantarimäki

Asiakas

Eija Mäntyharju

Johtoryhmä

Sanna Keskioja
Kimmo Simola

Kotisivu

`http://www.cs.helsinki.fi/group/karstula`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	23.4.2007	Dokumentti luotu
0.2	30.4.2007	Dokumenttia päivitetty
1.0	3.5.2007	Lopullinen dokumentti

Sisältö

1 Johdanto

Ylläpitodokumentti käsittelee sellaisia asioita, joita mahdolliset jatkoryhmät tai asiakkaan edustajat tarvitsevat muokatakseen ohjelmaa. Dokumenttia ei ole tarkoitettu luettavaksi yksinään, vaan yhdessä suunnitteludokumentin ja ohjelmakoodin kanssa. Näin ollen dokumentissa voi olla paljon viitteitä Meditrainer -projektin muihin dokumentteihin.

Ylläpitodokumentti jakaantuu kahdeksaan lukuun. Luvussa kaksi käydään läpi Meditrainer-ohjelman keskeinen sanasto. Kolmannessa luvussa esitellään ohjelman asennus- ja käynnistysohjeet, jotka löytyvät myös käyttöohjeesta. Luvussa 4 esitellään ne vaatimukset, jotka muuttuivat projektin kuluessa jäädytettyyn vaatimusmäärittelydokumenttiin verrattuna. Luvussa 5 käydään vastaavat muutokset läpi suunnitteludokumentin osalta. Luvussa kuusi esitellään ne ominaisuudet joita ei toteutettu vaatimusmäärittely - ja suunnitteludokumentista. Luvussa 7 käydään läpi ohjelman tekniseen ylläpitoon liittyviä seikkoja. Suunnitelman ylläpito -luvussa kerrotaan suunnittelun kohdat, jotka projektiryhmämme mielestä saattavat tarvita hiomista jatkossa, tai joihin voidaan jatkossa liittää uusia ominaisuuksia. Koodin ylläpito -luvussa käsitellään samat asiat koodin kannalta, jos on tarpeen. Testaus-luvussa käsitellään kohtia, jotka saattavat vaatia lisätestausta. Luvussa kerrotaan myös mahdollisista ongelmista, joita ei voitu syystä tai toisesta testata projektin aikana. Tietokanta-luvussa kerrotaan tietokannan rakenteeseen liittyviä korjaus- ja parannusehdotuksia.

2 Sanasto

Tässä luvussa määritellään ohjelman dokumentaatiossa käytetyt termit.

Desimaalilasku: Yksi laskutyypeistä. Desimaalilaskussa joko itse lasku tai sen lopputulos (tai molemmat) sisältävät desimaaliluvun.

Harjoittelujakso: Harjoittelujaksolla tarkoitetaan harjoitteluosiossa olevaa yhden laskutyypin yhtämittaista harjoittelua. Yksi harjoittelujakso kestää niin kauan, kunnes käyttäjä on vastannut oikein viiteen (oletusarvoisesti) laskuun ilman vihjeitä.

Harjoitteluosio: Harjoitteluosiolla tarkoitetaan ohjelman osaa, jossa käyttäjä voi harjoitella lääkelaskujen laskemista.

Harjoitteluosion pääsivu: Sivu, joka avautuu, kun valitaan ohjelman pääsivulta harjoitteluosio. Tällä sivulla käyttäjä voi valita mitä laskutyyppiä hän haluaa harjoitella.

Helppo lasku: Ylläpitäjä valitsee laskulle vaikeustason (helppo tai normaali) syöttäessään uutta harjoitteluosion laskua ylläpitosivun kautta. Helppo lasku tarkoittaa laskua, joka ei välttämättä ole sanallinen (kuten testissä) vaan voi olla esimerkiksi mekaaninen, numeroihin perustuva lasku (esim. Paljonko on 20 % luvusta 50).

Laskutyyppi: Laskut on jaettu neljään eri kategoriaan: murto-, desimaali-, prosentti- ja verrantolaskut. Näitä kutsutaan yhteisnimellä laskutyypit.

Läpipääsytodistus: Ohjelman tulostama paperi, josta ilmenee, että ohjelman käyttäjä on

päässyt testiosion läpi.

Lääkelasku: Lääkelaskulla tarkoitetaan kaikkia niitä laskutoimituksia, joita ohjelmalla lasketaan.

Murtolasku, murtolukulasku: Yksi laskutyypeistä. Murtolukulaskuissa joko itse lasku tai sen lopputulos (tai molemmat) sisältävät murtoluvun.

Normaali lasku: Ylläpitäjä valitsee laskulle vaikeustason (helppo tai normaali) syöttäessään uutta harjoitteluosion laskua ylläpitosivun kautta. Normaalilla laskulla tarkoitetaan laskua, joka on vastaava kuin testiosiossa. Normaalit laskut ovat aina sanallisia.

Ohjelman osio: Ohjelma sisältää neljä erillistä osiota: teoriaosio, harjoitteluosio, testiosio ja ylläpito-osio.

Ohjelman pääsivu: Sivu, joka avautuu, kun ohjelma käynnistyy. Pääsivulta on pääsy kaikkiin ohjelman osioihin.

Oppimateriaaliosio: Ks. teoriaosio.

Prosenttilasku: Yksi laskutyypeistä. Prosenttilaskuissa joko itse lasku tai sen lopputulos (tai molemmat) sisältävät prosenttiluvun.

Teoriaosio: Teoriaosiolla tarkoitetaan ohjelman osaa, jossa käyttäjä voi tutustua lääkelaskujen teoriaan.

Teoriaosion pääsivu: Sivu, joka avautuu, kun käyttäjä valitsee ohjelman pääsivulta teoriaosion. Teoriasivun pääsivulta on pääsy kaikkiin teoriadokumentteihin.

Testiosio = Koeosio: Testiosiolla tarkoitetaan ohjelman osaa, jossa käyttäjä voi tehdä lääkelaskutestin.

Verrantolasku: Yksi laskutyypeistä. Verrantolaskut muodostuvat suoraan ja kääntäen verrannollisuuksista.

Ylläpitonäyttö , ylläpidon pääsivu: Sivu, joka aukeaa, kun ylläpitäjä on valinnut ylläpitoosion ja syöttänyt salasanan oikein. Tältä sivulta ylläpitäjä voi valita haluamansa ylläpitotoiminnon.

Ylläpito-osio: Ylläpito-osiolla tarkoitetaan ohjelman osaa, jossa ylläpitäjä voi suorittaa ylläpitotehtäviä (salasanan vaihto, uusien laskujen lisääminen, vanhojen laskujen muokkaaminen ja poistaminen, läpipääsytodistuksen tulostaminen, yleiset ylläpitotehtävät).

3 Ohjelman asennus- ja käynnistysohje

3.1 Asennusohje

3.2 Asennusohje

Ohjelma toimitetaan asiakkaalle CD-levyllä ja asiakas vastaa ohjelman asennuksesta itse.

HUOM! Ohjelma kannattaa mieluiten asentaa Karstulan tiedostopalvelimelle siten, että

kaikilta koneilta on pääsy tähän ohjelmaan. Mikäli ohjelmaa ei asenneta tiedostopalvelimelle, tietyllä työasemalla lisätyt uudet/muokatut/poistetut kysymykset eivät näy muualle kuin kyseiselle työasemalle. Tämä tarkoittaa, että esimerkiksi uusi kysymys joudutaan lisäämään jokaiselle työasemalle erikseen. Tämä on hyvin työlästä mikäli koneita on useita. Mikäli ohjelma sijaitsee tiedostopalvelimella, niin se on asennettu sinne vain kertaalleen ja kaikki työasemat käyttävät samaa ohjelmaa. Tällöin myös kaikilta työasemilta tehdyt muutokset näkyvät muille koneille. Ylläpitotoimenpiteitä ei saa tehdä kuin yhdeltä koneelta kerrallaan, jos ohjelma on tiedostopalvelimella.

Jos ohjelma kuitenkin asennetaan jokaiselle koneelle erikseen, ylläpitotehtävät tulee tehdä aina yhdellä koneella. Ohjelma tallettaa tehdyt muutokset aina kyseisen koneen Meditrainer -kansioon test.xml ja practise.xml tiedostoihin. Tämä siis tarkoittaa sitä, että kun ylläpitotehtävät (esim. kysymysten muokkaus/lisäys/poisto) on tehty tietyllä koneella, niin ne ovat tallentuneet vain kyseisen koneen Meditrainer -kansioon test.xml ja practice.xml tiedostoihin. Tämän jälkeen nämä tiedostot pitää vielä kopioida esim. muistitikun avulla kaikille muille koneille näiden koneiden Meditrainer -kansioon. Näin muutokset siirtyvät myös muille koneille.

1. Laita CD tietokoneen CD- asemaan ja avaa sen sisältö.
2. CD sisältää useita eri kansioita, mutta asennuksessa tarvitaan vain Meditrainer nimistä kansiota.
3. Kopioi Meditrainer kansio kokonaisuudessaan haluamaasi paikkaan kotikoneellesi. Kopiointi Windowsissa onnistuu "vetämällä"(=drag and drop) kansio haluttuun hakemistoon koneelle. Vaihtoehtoisesti Meditrainer -kansio voidaan "kopioida ja liittää"haluttuun paikkaan. Kopioiminen tapahtuu klikkaamalla kansiota hiiren oikealla painikkeella ja valitsemalla listasta "Kopioi". Tämän jälkeen siirry haluttuun hakemistoon, klikkaa hiiren oikeaa painiketta ja valitse listasta "Liitä". Nyt ohjelma on kopioitu kotikoneellesi ja se on valmis käytettäväksi.
4. Voit luoda ohjelmasta myös pikakuvakkeen työpöydälle klikkaamalla meditrainer.exe-tiedostoa hiiren oikealla napilla ja valitsemalla "Lähetä- "Luo pikakuvake".

3.3 Käynnistysohje

1. Ohjelma käynnistetään samasta Meditrainer -kansiosta joka asennettaessa laitettiin kotikoneellesi. Mene siis aluksi kotikoneellasi Meditrainer -kansioon.
2. Meditrainer kansiossa on meditrainer.exe tiedosto. Ohjelma käynnistetään painamalla tätä kuvaketta kaksi kertaa hiiren vasemmalla painikkeella. Muista kansiossa olevista tiedostoista ei tarvitse välittää.
3. Jos olet luonut ohjelmasta pikakuvakkeen, ohjelma käynnistyy myös sitä klikkaamalla.

4 Jäädytyksen jälkeen muuttuneet vaatimukset

Projektiryhmämme pyrki jo määrittelyvaiheessa laatimaan vaatimukset siten, että niihin ei tule enää ainakaan suuria muuotoksia vaatimusmäärittelydokumentin jäädytyksen jälkeen. Tehdyt muutokset olivatkin pääasiassa joidenkin toimintojen tarkennuksia eivätkä niinkään suuria muutoksia. Alla olevassa listassa on käyty läpi nämä tehdyt muutokset/tarkennukset:

1. **Toiminnallisen vaatimuksen lisäys:** Määrittelydokumenttiin lisättiin toiminnallinen vaatimus, että ylläpitäjä voi muuttaa harjoitteluosion läpipääsyyn vaadittavien oikeiden vastausten lukumäärää ylläpito-osiosta. (Luku 5.1, vaatimus 27)
2. **Käyttäjän vastauksen muodon täsmentäminen:** Määrittelydokumenttiin täsmennettiin, missä muodossa käyttäjän vastaukset hyväksytään. Käyttäjän vastaukset hyväksytään joko murtolukuna, kokonaislukuna tai desimaalilukuna. (Luku 8.3)
3. **Virhetilanteiden lisääminen:** Määrittelydokumenttiin lisättiin neljä mahdollista uutta virhetilannetta (Luku 8.3, virhetilanteet 11-14)
4. **Käyttöliittymä protyyppikuvien päivitys:** Määrittelydokumenttiin päivitettiin uudet käyttöliittymäkuvat (Liite 1)
5. **Harjoittelussa vaadittavien vastausyritysten lukumäärä ennen kuin voi katsoa oikean vastauksen:** Päätettiin, että kun käyttäjä on vastannut kaksi kertaa väärin tai katsonut kaksi vihjettä, niin Ohita kysymys- ja Näytä vastaus -painikkeet aktivoituvat. Alkuperäisessä määrittelydokumentissa oli määritetty, että käyttäjältä vaaditaan kolme vastausyritystä.

5 Jäädytyksen jälkeen muuttuneet suunnitelman osat

Toteutettu järjestelmä muuttui jonkin verran suunnitteludokumenttiin nähden. Eniten muutoksia tuli käyttöliittymän toteutuksen yhteydessä. Tässä luvussa on lueteltu järjestelmän muuttuneet osat. Luokkien API-kuvauksista voi myös nähdä tehdyt muutokset. HTML-muotoiset API-kuvaukset löytyvät asennus-CD:ltä tiedostosta api/index.html.

Korkean tason muutos järjestelmässä oli se, että osa tarkkailijamallin tarkkailtavan roolista siirrettiin logiikkakomponentilta käyttöliittymän uudelle *GuiHandler*-luokalle. Tämä selitetään tarkemmin *GuiHandler*ia käsittelevässä kohdassa. Suunnitelmassa ei myöskään määritelty tarkemmin, miten tarkkailijamalli tultaisiin toteuttamaan. Se päädyttiin toteuttamaan Javan standardikirjaston välineillä: *Observable*-luokalla ja *Observer*-rajapinnalla. Tarkkailtavat perivät *Observable*-luokan ja tarkkailijat toteuttavat *Observer*-rajapinnan.

Osa tarkkailtavan roolista jäi logiikkakomponentille: kysymystilastojen päivittymisen seuraaminen. Suunnitteludokumentissa ei otettu tarkemmin kantaa siihen, miten logiikkakomponentti hoitaisi tarkkailtavan roolin. Toteutuksessa päädyttiin siihen, että logiikkakomponentin *Practice*-luokka huolehtii tarkkailtavan roolista kysymystilastojen päivittymisessä. Muuten logiikkakomponentti ei tarkkailijamallia käytä.

5.1 Käyttöliittymän muuttuneet osat

Käyttöliittymän muuttuneet ja lisätyt metodit sekä kentät selviävät API-kuvauksista, jotka löytyvät asennus-CD:ltä tiedostosta api/index.html. Tämän dokumentin liitteenä on päivitetty kuva käyttöliittymäkomponentin luokkakaaviosta.

5.1.1 GuiHandler

Käyttöliittymäkomponenttiin luotiin uusi luokka *GuiHandler*. Suunnitteludokumentissa sanottiin, että ohjelma huolehtii tarkkailijamallin avulla siitä, mikä käyttöliittymän näkymä on kulloinkin näkyvissä, ja että logiikkakomponentti hoitaa tässä tarkkailtavan roolin. Tämä rooli otettiin logiikkakomponentilta pois, ja luokka GuiHandler luotiin huolehtimaan tästä asiasta. Lisäksi GuiHandler huolehtii myös käyttöliittymän ikkunoiden luomisesta ohjelmaa käynnistettäessä. Logiikkakomponentti ei enää ota kantaa käyttöliittymään, esim. siihen, mikä näkymä näkyy milloinkin, tai näkyvätkö vaikka kaikki näkymät samaan aikaan. Logiikkakomponenttia voi näin käyttää erilaisten käyttöliittymien kanssa.

5.1.2 Pääohjelma ja ohjelman käynnistys

Käyttöliittymäkomponenttiin luotiin uusi luokka *Meditrainer.java*, joka luo logiikkatason oliot ja GuiHandlerin, jolle välittää logiikkaoliot. GuiHandler puolestaan luo ali-ikkunat ja välittää logiikkaoliot niille.

Järjestelmän pääikkuna *ParentView* on ainoa luokka, joka periytyy luokasta *JFrame*. Loput ikkunat perityvät luokasta JInternalFrame ja ne lisätään pääikkunan alle. Pääikkunaa lukuunottamatta luokat rekisteröidään GuiHandlerin tarkkailijoiksi siten, että ne toteuttavat rajapinnan *Observer*. GuiHandler-luokka säilyttää viitteet kaikkiin ali-ikkunoihin ja pitää kirjaa siitä, mikä ikkuna on näkyvissä. Kun jokin ikkuna pyytää toista ikkunaa näkyville, GuiHandler kutsuu kaikkien rekisteröityneiden tarkkailijoittensa update-metodia. Tässä voidaan välittää myös parametrina tieto näkyville halutusta kategoriasta muodossa *Question.Category*.

5.1.3 ParentView

Toteutusvaiheessa ikkunanhallinta ja viitteet siirtyneet siirtyivät *ParentView*ltä *GuiHandlerille*. Luokkaan lisättiin ominaisuus ominaisuus, jonka avulla se hoitaa myös ohjelmaa suljettaessa varmistusviestien näyttämisen.

5.1.4 MainView

Luokkaan lisättiin Tulessaan näkyville *MainView* tyhjentää salasanakentät, nollaa harjoitussarjan ja päivittää tilastotiedot. Tarkkailijamallin mukaisesti tilastotiedot haetaan logiikan *Practice* - oliolta. Siirryttäessä kokeeseen tai ylläpitoon salasana tarkastetaan logiikan *Guard* - oliolta.

5.1.5 TheoryView

Toteutuneessa järjestelmässä ei tarkisteta onko käyttäjä samanaikaisesti sekä harjoittelutilassa että teoriaosiota selailemassa.

5.1.6 TestView

Luokan konstruktori ottaa viitteet *ParentViewin* lisäksi myös luokkiin *TestFactory* ja *Print*. Metodit ovat erilaisia kuin suunnitteludokumentin luokkakaaviossa kuvatut, mutta ne ovatkin lähinnä TestViewin sisäistä toimintaa. Ulospäin testViewistä näkyvät ainoastaan konstruktori ja tarkkailijamallin update-metodi.

5.1.7 PracticeView

Toteutuneessa järjestelmässä *PracticeView* ei käytä tarkkailijamallia hakiessaan kysymystilastoja Practice-luokalta).

5.1.8 AdminView

Luokalle *AdminView* lisättiin apuluokat *AdminPasswordView*, *AdminPrintView*, *AdminSettingsView*, *QuestionTableModel* sekä *TableSorter*.

5.2 Logiikkakomponentin muuttuneet osat

5.2.1 Practice

Metodi *checkAnswer* heittää *IllegalStateException* - poikkeuksen, jos uutta kysymystä ei ole haettu. Metodilla voi toteutuneessa versiossa tarkistaa saman kysymyksen moneen kertaan, poikkeus heitetään vain, jos kysymystä ei ole haettu ollenkaan. Tilastoja päivitetään tietenkin vain ensimmäisen vastauksen perusteella.

Metodi *getNextQuestion* ei heitä tarkistettua *NotEnoughQuestionsException* - poikkeusta kysymysten loppuessa, vaan heittää tarkistamattoman *IllegalStateException* - poikkeuksen, kun harjoittelusarja on valmis. Sarja on valmis myös silloin, kun kysymykset loppuvat. Jos helpot kysymykset loppuvat, ei heitetä poikkeusta, vaan siirrytään suoraan normaaleihin.

Metodi *isPracticeActive* poistetttiin.

Metodit *getNextQuestion* ja *endPractice* heittävät IllegalStateExceptionin, jos niitä kutsutaan ennen kuin kysymyskategoria on valittu.

Metodi *notifyHintUsed* heittää emphIllegalStateException - poikkeuksen, jos kysymystä ei ole vielä haettu.

Metodi *endPractice* alustaa suoraan uuden kysymyssarjan ja heittää siis emphNotEnoughQuestionsException - poikkeuksen, jos kysymyksiä ei ole tarpeeksi. Tällöin metodi ei tee mitään.

Metodi *changeCategory* heittää *NotEnoughQuestionsException* - poikkeuksen, jos uudessa kategoriassa ei ole tarpeeksi kysymyksiä. Tällöin metodi ei tee mitään.

Metodi *getStatistics* - metodi ottaa parametrina kysymyskategorian, jonka tilastotiedot se palauttaa. *Statistics*-olio sisältää ainoastaan yhden kategorian tiedot.

Jotkin metodit heittävät *DBLogicException* - poikkeuksen, vaikka suunnitteludokumentissa ei ole sitä mainittu.

Practice-luokka toteuttaa tarkkailijamallin mukaisen tarkkailtavan roolin perimällä Javan standardikirjaston *Observable*-luokan. Sitä voi tarkkailla kysymystilastojen päivittymisestä. Suunnitteludokumentissa ei määritelty tarkemmin, miten logiikkakomponentti hoitaisi tämän asian. Tästä on kerrottu tarkemmin aiemmin tässä ylläpitodokumentissa.

5.2.2 Maintenance

Luokan konstruktori ottaa parametrina myös viitteen *Guard* - olioon.

Metodi *getQuestionObject* muutettiin *getUnusedQuestionID* - metodiksi, sillä *Question* - luokka muutettiin muuttamattomaksi.

Jotkin metodit heittävät *DBLogicException* - poikkeuksen, vaikka suunnitteludokumentissa ei ole sitä mainittu.

ylläpitäjän oikeudet vaativat metodit heittävät *NotAdminException* - poikkeuksen, jos ylläpitäjä ei ole kirjautunut ohjelmaan.

Luokkaan lisättiin useita metodeita. Metodi *saveSettings* joka tallentaa teoriatiedostojen polun,kysymystiedostojen polun, todistusten tulostusmäärän ja oletustulostimen nimen. Metodit *getDefaultPrinter* ja *getCertificateCopies* palauttavat oletustulostimen nimen ja tulostettavien todistusten lukumäärän. Kysymyssarjan kysymysten lukumäärän sekä teoria- ja kysymystiedostojen polun palauttavia metodeita lisättiin *getSetCount*, *getTheoryPath* sekä *getQuestionPath*. Metodi *changeDistribution* nimettiin udelleen *saveSecretSettings* - nimiseksi metodiksi.

5.2.3 Test

Luokan konstruktori ottaa parametrina myös viitteen *Guard* - olioon. Konstruktori myös heittää *NotAdminException* - poikkeuksen, jos ylläpitäjä ei ole kirjautunut ohjelmaan.

Metodit *checkPass* ja *checkAnswers* heittävät *IllegalArgumentException* - poikkeuksen, jos niille annetaan eri määrä vastauksia kuin testissä on kysymyksiä.

5.2.4 Print

Toteutuneessa järjestelmässä *print*-luokka ei ole staattinen, vaan siitä luodaan ilmentymä, ja ilmentymälle annetaan parametrina viite *DBHandleen*.

Tulostaminen hoidetaan metodilla *PrintTestCertificate*. Sille annetaan parametrina sekä nimi että henkilötunnus. Sille voi antaa parametrina joko viitteen tietokantaan, jolloin se hakee tulostinasetukset sieltä, tai suoraan oletustulostimen nimen ja kopioiden määrän. Jos metodi epäonnistuu tulostinasetuksien haussa, heitetään *DBLogicException* - poikkeuks. Jos oletustulostinta ei löydy, tai sen arvoksi on annettu null, tulostus toimii, mutta jättää oletustulostinasetuksen huomiotta.

getPrinterNames - metodin nimi muutettiin täsmällisemmäksi nimeksi *getPrintServiceNames*.

5.2.5 Theory

Luokkaan lisättiin metodi getConversion, joka palauttaa yhksikkömuunnosten teoriatekstin.

5.2.6 MathTools

MathTools on *logic*-paketin sisäinen apuluokka, joka sisältää permutointimetodeja. Näillä arvotaan kysyttäviä kysymyksiä ja niiden järjestystä.

5.2.7 Answer

Answer-luokassa on *toString*-metodi, joka tulostaa vastauksen siinä muodossa (desimaali tai murtoluku), jossa se on annettu. *Answer*-luokka toteuttaa myös Javan *equals*- ja *hashCode*-metodit. Metodin *hashCode* se toteuttaa siksi, että sen toteuttaminen on hyvää tyyliä, jos toteutetaan myös metodi *equals*.

Answer-luokka siirrettiin *meditrainer*-pakkauksen juureen.

5.2.8 TestFactory

TestFactory on *logic-paketin* luokka, joka on *Test*-luokan luomista varten.

5.2.9 GuiTools

Käyttöliittymäpakkaukseen on lisättiin *GuiTools* - luokka, joka hoitaa käyttäjälle näytettävät virheilmoitukset.

5.2.10 Question

Question-luokka siirrettiin *meditrainer*-pakkauksen juureen.

5.3 Tietokantakomponentin muuttuneet osat

5.3.1 XMLParser

Luokalla *XMLParser* ei ole konstruktoria. Luokan metodit muutettiin staattisiksi, koska se on kirjasto.

5.3.2 Crypt

Luokasta poistettiin tarpeettomana metodi *checkPassword*.

5.3.3 QuestionDB

Luokkaan lisättiin metodit *clear*, *getQuestion* ja *shallowCopy*. Viimeksi mainittu metodi lisättiin myös luokkiin *SettingsDB* ja *SecretSettingsDB*. Metodia tarvitaan, jotta saadaan säilytettyä samat viitteet *DBHandlessa*.

5.3.4 TheoryDB

Yksikkömuunnosten tullessa mukaan teoriaosioon *TheoryDB*-luokkaan lisättiin *String*-tyyppinen kenttä *conversion*.

6 Toteutumattomat vaatimukset

Vaatimusmäärittelyssä mainittu vaatimus 26 jätettiin toteuttamatta. Vaatimus oli määritelty alhaisimmalla prioriteetilta.

7 Ylläpitoon liittyvät seikat

7.1 Suunnitelma

Koska ohjelmistotuotantoprojektin aikataulu oli suhteellisen tiukka, ohjelma suunniteltiin toiminnallisuudeltaan melko yksinkertaiseksi. Vaatimusmäärittely- ja suunnitteluvaiheessa pyrittiin rajaamaan pois ominaisuuksia, jotka olisivat olleet liian raskaita toteuttaa. Seuraavassa muutamia ajatuksia, jotka suunnitteluvaiheessa tulivat esille, mutta jätettiin pois:

- Eräs asiakkaan toive ohjelman mahdollisuudeksi ominaisuudeksi oli käyttäjän vastauksen analysoiminen ja palautteen antaminen tämän perusteella (palaute voisi olla esimerkiksi, missä kohdassa käyttäjä todennäköisesti teki virheen, jos hän antoi väärän vastauksen). Tämä todettiin sen verran vaikeasti toteutettavaksi ominaisuudeksi, että sitä ei otettu projektiin mukaan. Tämän lisääminen ohjelmaan olisi yksi parannusmahdollisuus, ja tekoälyn tuntemus auttanee sen toteuttamisessa.

- Ohjelmaa voisi jatkokehittää siten, että siinä käytettävät laskut olisi mahdollista sisällyttää useampaan kuin yhteen osa-alueeseen. Sama lasku voi sisältää mm. desimaalilaskun ja verrantolaskun ominaisuuksia. Ohjelman tulisi silti valvoa, ettei samaa laskua kysyttäisi useaan kertaan harjoitusjakson tai kokeen aikana.

- Harjoitusosioon voisi lisätä komponentteja, jotka selkeyttävät harjoitusjakson tilannetta. Tällaisia voisivat olla mm. pylväsdiagrammit, jotka kuvaavat oikein ja väärin vastattujen kysymysten määrää harjoitushetkellä.

- Jos ohjelma leviää laajempaan käyttöön, toinen mahdollinen lisäys ohjelmaan olisi lokalisaatiotuki. Sen lisäämisessä on jonkin verran työtä, sillä ohjelman tekstit ja ilmoitukset on nyt kovakoodattu ohjelmakoodin sekaan.

- Yksi muutosmahdollisuus olisi myös ohjelman käyttöliittymän muuttaminen verkon yli web-selaimella toimivaksi. Tämä olisi melko suuri muutos, mutta ohjelman modulaarisen rakenteen vuoksi osaa nykyisen ohjelman komponenteista saattaisi voida käyttää siinä muuttamatta tai pienin muutoksin.

7.2 Koodi

Koko ohjelma on toteutettu Java-kielellä, ja kirjastoina on käytetty ainoastaan Javan standardikirjastoja. Ohjelmassa on käytetty Javan versio 5:en tuomia uusia ominaisuuksia kuten geneerisyyttä.

Ohjelman rakenne on melko modulaarinen ja koodi aika suurelta osin yleisyyteen pyrkivää, joten uusien ominaisuuksien lisäämisen pitäisi olla helppoa. Kaikilta osin nämä ominaisuudet eivät kuitenkaan toteudu. Esimerkiksi joissakin kohdissa ohjelmaa, joissa tarvitsee käydä kaikki kysymyskategoriat läpi, ne käydään läpi yleisellä kategoriavalikoimalla toimivassa silmukassa. Nämä kohdat toimivat sellaisinaan, vaikka ohjelmaan lisättäisiin uusi kysymyskategoria tai vanhoja kategorioita muutettaisiin. Joissakin kohdissa kuitenkin kategorioita käsitellään suoraan niiden nimillä, joten näiden kohtien muuttaminen vaatii enemmän työtä. Eräs parannusmahdollisuus koodiin olisikin tällaisen yleisyyden lisääminen.

Projektiryhmässä pyrittiin jossain määrin noudattamaan yhteisiä ohjelmointikäytäntöjä: koodin ulkoasun osalta nämä noudattelevat pitkälti Sunin suosituksia. Koska ohjelman eri osat ovat kuitenkin eri ihmisten tekemiä, ylläpitäjän kannattaa varautua siihen, että ohjelman eri osat on tehty jossakin määrin eri tyyleillä. Mitään erityisiä ohjelmointitekniikoita tai suunnittelumalleja ei käyttöliittymän tarkkailijamallin ja tietokannan julkisivumallin lisäksi käytetty, mutta toteutuksessa pyrittiin yleiseen modulaarisuuteen, siisteyteen ja

hyviin ohjelmointitapoihin. Koodi pyrkii olemaan luettavaa sekä sisältämään kommentteja sopivissa kohdissa, joten sen toiminnasta pitäisi yleensä saada selvä sitä lukemalla. Sen vuoksi tässä luvussa ei selitetä koodin toimintaa tarkemmin.

Suurempia ongelmia ei toteuttamisessa ollut, ja niiden ilmenemiseen tuskin on suurta syytä varautua ohjelman ylläpidossakaan.

7.3 Testaus

Testauksen tavoitteena oli 90 prosentin lausekattavuus ja täysin automatisoitu testaus. JUnitilla ja Jemmyllä automatisoidut testit kattavat gui-, logic- ja db-komponenteista 42, 67, ja 85 prosenttia. Käyttöliittymän laajempi testaus Jemmyllä olisi suotavaa, koska se mahdollistaisi helposti järjestelmän rasitustestauksen. Silloin voisi esimerkiksi ajaa käyttötapauksia lukuisia kertoja läpi eri järjestyksissä ja löytää tilanteita, joissa järjestelmä joutuu sisäisesti virheelliseen tilaan tai vuotaa muistia. Tulostusta ei ole testattu kunnolla.

7.4 Tietokanta

Tietokanta on järjestelmän alin komponentti, jonka tarkoitus on säilyttää testi- ja harjoituskysymyksiä. Sen toteutukseen harkittiin kolmea vaihtoehtoa. Relaatiotietokantaa ei voitu valita, koska asiakkaan käytössä olevista palvelimista ei ollut tietoa. Toinen vaihtoehto oli paikallinen SQLite-tietokanta. Sitä ei voitu käyttää, koska se ei tarjoa salausmahdollisuutta ilman lisäinvestointia. Lopulta päädyttiin yksinkertaiseen XML-tietokantaratkaisuun. Myös järjestelmän asetukset tallennetaan XML-muotoon. XML-muotoisia kysymys- ja asetustiedostoja on helppo laajentaa tarvittaessa. Niiden huonona puolena on skaalautuvuus. Jos XML-tiedostot kasvavat suuriksi, niiden käyttö järjestelmässä hidastuu, koska ne parsitaan aina kokonaan muistiin ja generoidaan kokonaisina levylle. Lähes kaikki toteutetun kysymystietokannan hakumetodeista ovat luokkaa O(n). Jos tästä muodostuu pullonkaula, hakupuut tai vastaavat ovat vaihtoehtoja.

8 Havaitut virheet

Ohjelmaa testattiin sen koko toteutusvaiheen ajan. Virheitä pyrittiin korjaamaan sitä mukaa, kun niitä havaittiin. Tässä kappaleessa on lueteltu virheitä ja puutteellisia ominaisuuksia, jotka jäävät toteutuneeseen ohjelmaan. Peruskäyttäjän kannalta ohjelman katsotaan silti toimivan riittävän hyvin.

- Tulostus ei aina toimi ainakaan Linux-ympäristössä. Tämä on todennäköisesti Java-ympäristön oma virhe. Tulostus ei anna virheilmoitusta, mutta ei myöskään tulosta mitään kunnollista.
- Ylläpitonäkymässä ei tarkisteta, että kokeeseen valittu kysymysjakauma ei ylitä tietokannassa olevaa koekysymysten määrää. Tämä ei ole varsinaisesti virhe, vaan puuttuva ominaisuus.

- Jos käyttäjä ei läpäise koetta, hänelle näytetään kysymykset, jotka menivät väärin ja näiden kysymysten tiedot. Jos nämä kysymykset, niiden yksiköt, laskukaavat tai vastausten tekstimuodot (käyttäjän antamat tai kysymykseen sisältyvät) sisältävät (tarkoituksettomasti) HTML-muotoilua, niin ne näkyvät tässä näytössä väärin, sillä ne näytetään HTML-muotoilua käyttävässä paneelissa.

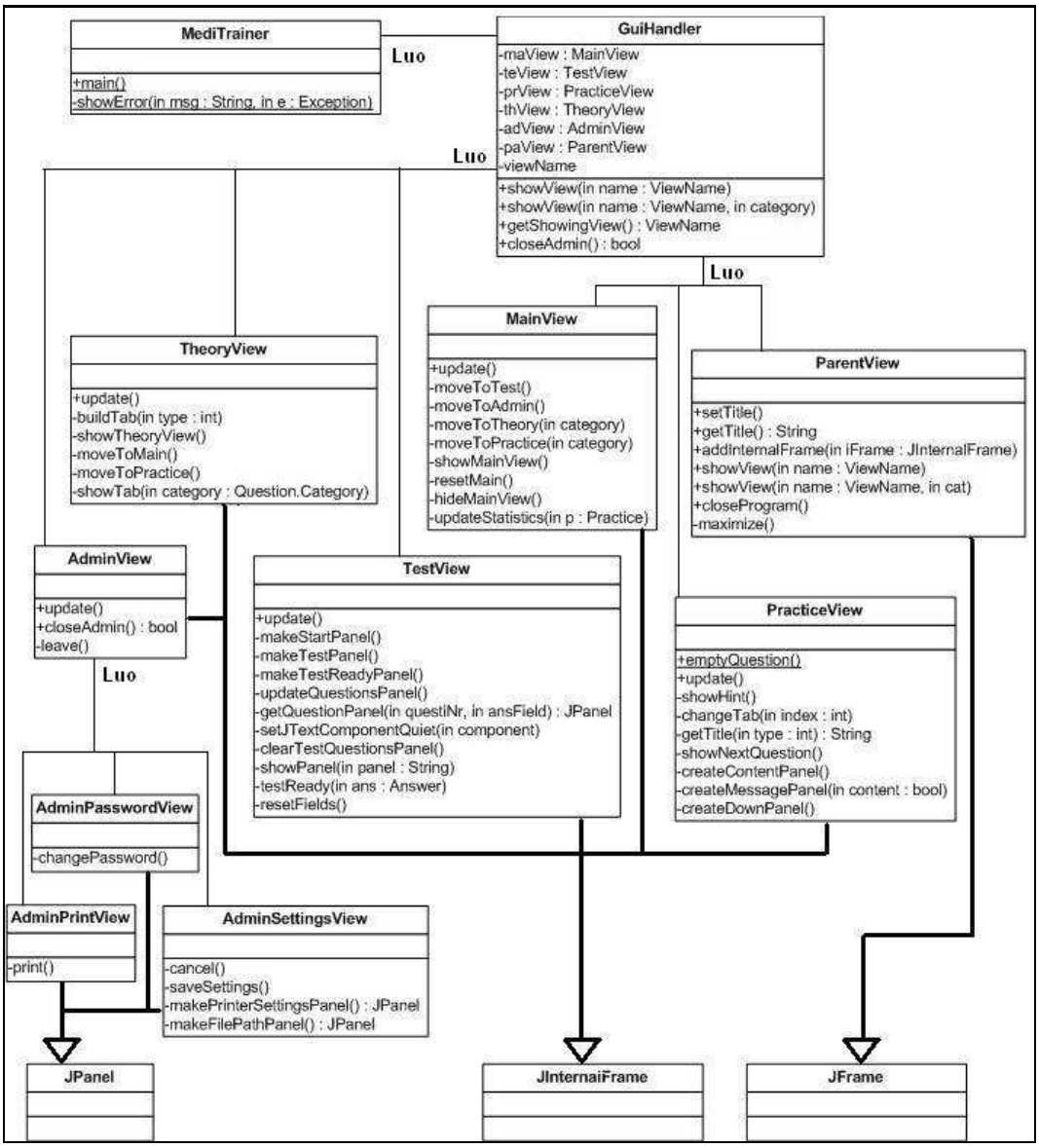
- Joskus havaittiin, että ohjelman sulkeutuessa tekstikonsoliin, josta ohjelma käynnistettiin, tulostuu virhe tai konsoli jäi jumiin

- Ohjelman käynnistyessä päänäkymän sisältö liikkuu jonkin verran, ilmeisesti siksi, että näkymä näkyy jo silloin, kun siihen ollaan vielä lisäämässä sisältöä.

- Ohjelman ali-ikkunoissa on otsikkopalkki, joka on tavallisen ikkunan otsikkopalkin näköinen (ulkonäkö riippuu toki siitä, mitä Javan "Look and Feel- tyyliä käytetään). Otsikkopalkin voisi poistaa kokonaan ja laittaa vaihtuvan otsikon pääikkunaan.

- Ikkuna ei ole aina maksimikokoinen, ainakaan, kun ohjelma käynnistetään jar-paketista.

Liite 1. Käyttöliittymäkomponentin luokkakaavio



Kuva 1: Käyttöliittymäkomponentin päivitetty luokkakaavio