Kasino

Ohjelmointistudio 2: Projekti

Dokumentti

Joona Lillfors

648080

Tietotekniikka, Perustieteiden korkeakoulu

Vuosikurssi 2017

26.4.2018

Yleiskuvaus

Kasino on korttipeli, jossa kerätään pisteitä, jotka lasketaan jokaisen kierroksen lopussa. Peli jatkuu, kunnes joku pelaajista saavuttaa 16 pistettä. Jokaisen pelikierroksen alussa korttipakka sekoitetaan ja jakaja jakaa jokaiselle pelaajalle 4 korttia, jotka eivät näy muille kuin pelaajalle itselleen, sekä 4 korttia pöytään, jotka näkyvät kaikille pelaajille. Loput kortit jätetään pöydälle pinoon ylösalaisin. Jakajasta seuraava aloittaa pelikierroksen, seuraavalla kierroksella hän on jakaja.

Pelaaja voi kerrallaan käyttää jonkin kädessään olevista korteista, joko ottaa sillä pöydästä kortteja tai laittaa kortin pöytään. Jos pelaaja ei voi ottaa mitään pöydästä täytyy hänen laittaa jokin korteistaan pöytään. Pelaajan ottaessa pöydästä kortteja hän kerää ne itselleen pinoon. Pinon sisällöstä lasketaan korttien loputtua pelaajan pisteet. Pöydässä olevien korttien määrä voi vaihdella vapaasti. Jos pöydässä ei ole yhtään korttia, täytyy seuraavan pelaajan laittaa jokin korteistaan tyhjään pöytään. Aina käytettyään kortin pelaaja ottaa käteensä pakasta uuden, niin että kädessä on aina 4 korttia. Kun pöydällä oleva pakka loppuu, ei oteta enää lisää vaan pelataan niin kauan kuin kenelläkään on kortteja kädessä. Kortilla voi ottaa pöydästä yhden tai useampia samanarvoisia kortteja ja kortteja, joiden summa on yhtä suuri, kuin kortin jolla otetaan.

Kun kaikilta loppuvat kädestä kortit saa viimeksi pöydästä kortteja ottanut loput pöydässä olevat kortit. Tämän jälkeen lasketaan pisteet ja lisätään ne entisiin pisteisiin. Pelissä on muutama kortti, joiden arvo on kädessä arvokkaampi kuin pöydässä, ässät ovat kädessä 14 ja pöydässä 1, ruutu-10 on kädessä 16 ja pöydässä 10, pata-2 on kädessä 15 ja pöydässä 2. Muiden korttien arvot ovat samat sekä pöydässä että kädessä. Jos joku saa kerralla pöydästä kaikki kortit kerralla, hän saa ns. mökin, joka merkitään muistiin. Pistelasku menee seuraavasti. Jokaisesta mökistä ja ässästä saa yhden pisteen. Eniten kortteja saanut saa yhden pisteen ja eniten patoja saanut saa 2 pistettä. Ruutu-10 kortin omistaja saa 2 pistettä ja pata-2 kortin omistaja saa yhden pisteen.

Peliä voi pelata 2-8 käyttäjää ja lisäksi peliin voidaan lisätä 0-8 kappaletta tietokonevastustajia. Työ on toteutettu vaativana.

Käyttöohje

Ohjelma käynnistetään ajamalla scala-tiedosto ”Kayttoliittyma.scala”. Käyttäjän käynnistäessä ohjelman aukeaa pelin aloitusruutu. Aloitusruudulla voi valita joko uuden pelin tai ladattavan viimeksi tallennetun pelin tiedostosta. Jos käyttäjä valitsee uuden pelin, tulee ruutu, jossa lisätään peliin pelaajat. Ruudulla on valinnat ”Reset”, joka poistaa lisätyt pelaajat, napit ”Tietokonevastustaja” ja ”Pelaaja”, joista valitaan toinen kuvaamaan pelaajan tyyppiä, ”Lisää pelaaja”, joka lisää pelaajan, jos tekstikentällä on tekstiä ja ”Aloita peli”, joka aloittaa pelin, jos käyttäjä on lisännyt vähintään 2 pelaajaa. Lisäksi ”Lisää pelaaja”-painikkeen yllä on tekstikenttä, johon syötetään lisättävän pelaajan nimi. Lisättyjen pelaajien nimet näkyvät ruudun vasemmassa ylälaidassa. Jos aloitusehdot täyttyvät ”Aloita peli”-painike vie käyttäjän pelitilanteeseen. Aloitusruudun ”Lataa peli”-painikkeella päästään suoraan pelitilanteeseen tai lopetusruudulle, jos tallennettu peli on ohitse.

Pelitilanteessa ruudulla näkyy ylhäällä pöydällä olevat kortit ja alhaalla vuorossa olevan pelaajan kädessä olevat kortit, jos kyseessä on Pelaaja, Tietokonevastustajan tapauksessa kädessä olevia kortteja ei näy. Alimpana painikkeet ”Tietokoneen siirto” ja ”Ota/laita kortti”. Jos vuorossa on Tietokonevastustaja, ”Tietokoneen siirto”-painiketta painamalla Tietokonevastustaja tekee siirtonsa ja vuoro siirtyy seuraavalle, muuten painike ei tee mitään. Käyttäjä voi valita kädessä olevista korteista kerrallaan yhden korttia painamalla, jolloin kortin reunus tummentuu. Vastaavasti pöydällä olevista korteista voi valita useampia. Ollessa Pelaajan vuoro käyttäjä voi painaa ”Ota/laita kortti”-painiketta tehdäkseen siirron. Jos siirto on laiton tai kortteja ei ole valittu, ei tapahdu mitään.

Kun joku pelaajista on saavuttanut 16 pistettä pelikierroksen lopussa, päästään voittoruudulle. Voittoruudulla lukee pelin voittaneen pelaajan nimi, sekä jokainen pelissä ollut pelaaja ja heidän saamansa pisteet.

Joka ruudulla on valikkopalkki, josta pääsee aloitusruudulle painikkeella ”Alkuun”, voi tallentaa pelitilanteen painikkeella ”Tallenna peli”, jos ruudulla on käynnissä oleva pelitilanne tai voittoruutu, muuten painike ei tee mitään ja ”Sulje”-painike, joka sulkee ohjelman.

Ohjelman rakenne

Ohjelmalla on luokat Pelikierros, Kortti, Pelaaja, Tietokonevastustaja ja Käyttöliittymä, sekä objekti Peli. Luokka Pelikierros kuvaa pelitilannetta pelaajien vuorojen välissä. Se sisältää tiedot pelissä olevista pelaajista, kierroksen jakajasta, pöydällä olevista korteista, korttipakassa olevista korteista ja kenen pelaajan vuoro on. Lisäksi sen metodeilla voidaan laskea ja lisätä pelaajien pisteet kierroksen lopuksi, nostaa korttipakasta kortti, katsoa onko korttipakka tyhjä ja tallentaa pelitilanne tiedostoon.

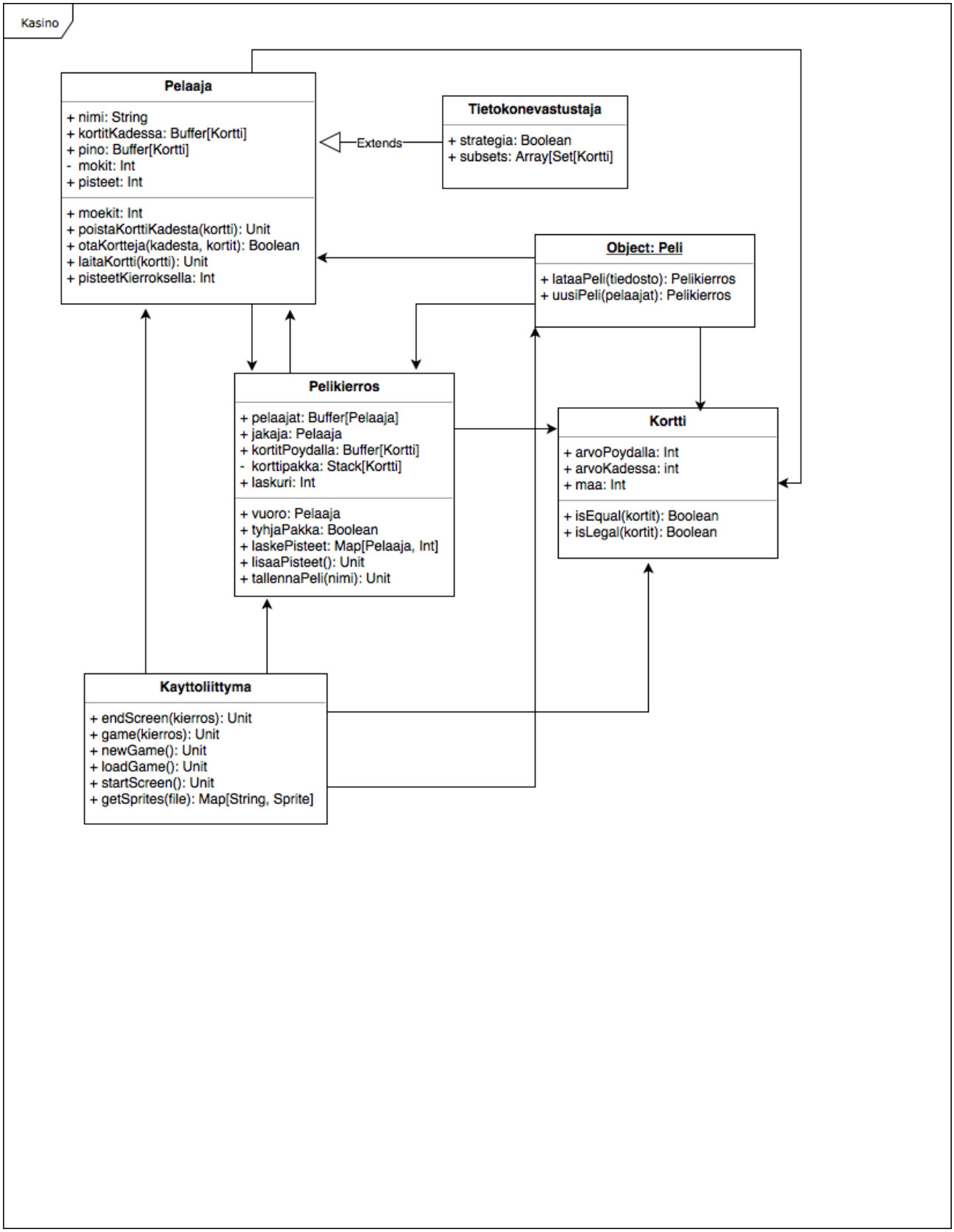
Luokka Kortti kuvaa yksittäistä korttia. Se sisältää tiedon kortin arvosta pöydällä, pelaajan kädessä ja kortin maan(risti, pata, hertta tai ruutu). Sen metodeilla voidaan katsoa, onko kyseisen kortin arvo kädessä yhtä suuri kuin metodille parametrina annettujen korttien arvojen pöydällä summa. Lisäksi voidaan katsoa, voidaanko kyseisellä kortilla ottaa pöydästä metodille parametrina annetut kortit.

Luokka Pelaaja kuvaa yksittäistä pelaajaa. Se sisältää tiedot pelikierroksesta, pelaajan pinossa olevista korteista, kädessä olevista korteista, pelaajan nimen, pelaajan mökit ja pisteet. Lisäksi luokan metodeilla voidaan laittaa pelaajan kädestä kortti pöytään ja ottaa kortteja pöydästä, sekä nostaa aina uusi kortti pakasta käytettyään kädessä olevan kortin. Voidaan myös laskea pelaajan mökeistä, ässistä ja erikoiskorteista saadut pisteet kierroksella. Luokka Tietokonevastustaja perii luokan Pelaaja. Se sisältää metodin, joka tekee tietokonevastustajan siirron vuoron aikana.

Objekti Peli sisältää pelitilanteen luomiseen ja lataamiseen tiedostosta liittyviä metodeja, jotka eivät oikein sopineet muihin luokkiin. Metodit joko lataavat pelattavaksi keskeneräisen pelin tai alustaa uuden pelin pelattavaksi.

Käyttöliittymä käyttää metodeja, jotka kuvaavat tiettyä tilaa ohjelmassa. Näitä tiloja ovat aloitus, uuden pelin alustus, pelin lataus, itse pelitilanne, sekä pelin päättyminen. Näitä tiloja vastaavat metodit piirtävät ruudulle tilan ja kutsuvat uuden tilan metodia tilan muuttuessa.

Lähes kaikki luokat ovat suhteessa toisiinsa kuten näkyy UML-kaaviosta kuvasta 1.



Kuva1: UML-kaavio ja luokkien väliset suhteet

Algoritmit

Keskeisimpiä algoritmeja ovat pöydästä otettujen korttien tarkistusalgoritmi ja tietokonevastustajan strategia. Korttien tarkistusalgoritmi toimii siten että se aluksi muodostaa annetuista korteista kaikki osajoukot joiden summat arvoista pöydällä on yhtä suuria kuin kädessä olevan kortin arvo. Jos yhtään osajoukkoja ei ole, siirtoa ei hyväksytä. Muutoin etsitään, onko olemassa sellaista osajoukkojen kombinaatiota, jossa olisi käytetty kaikkia annettuja kortteja tasan kerran.

Tietokonevastustajan algoritmi katsoo kaikki mahdolliset siirrot osajoukkojen avulla. Jos mahdollisia siirtoja ei ole, se laittaa pöydälle sellaisen kortin minkä arvoisia kortteja sillä on useampi ja jos tällaisia kortteja ei ole, se laittaa kädessä olevasta arvoista pienimmän pöytään. Jos mahdollisia siirtoja taas on olemassa, tietokone valitsee sellaisen, millä saa eniten kortteja nostettua pöydästä.

Tietorakenteet

Luokassa Pelikierros tarvitaan tietorakenteita eri pelaajien, pöydällä olevien korttien ja pakassa olevien korttien tiedossa pitämiseen. Pelaajat ovat Vector-taulukossa, ettei pelissä olevien pelaajien määrää voisi vaihtaa kesken kaiken, eikä sen muuttaminen olekaan tarpeellista. Pöydällä olevat kortit voidaan pitää Buffer-taulukossa, koska pöydällä voi käytännössä olla kuinka monta korttia tahansa. Korttipakan toteutukseen sopii pinorakenne, eli pinosta voidaan poistaa vain päällimmäinen kortti kerrallaan. Näin korttipakan on tarkoituskin toimia tässä pelissä.

Luokassa Pelaaja on pelaajan pino kortteja ja pelaajan kädessä olevat kortit. Korttipino voidaan toteuttaa Buffer-taulukoa käyttäen, koska pinossa on aina vaihteleva määrä kortteja. Pelaajan kädessä olevat kortit voidaan varastoida Buffer-taulukkoon. Buffer sopii tähän koska kädessä olevien korttien määrä vaihtelee.

Listojen sijasta olisi voitu käyttää Set-rakenteita. Kuitenkin tuntui selkeämmältä käyttää listoja, koska niissä pidetään yllä järjestystä, toisin kun Set-rakenteissa. Lähes kaikki käytetyt tietorakenteet ovat muuttuvatilaisia (mutable), koska se mahdollistaa listojen helpon muokkauksen. Jos oltaisiin valittu noudatettavan funktionaalista ohjelmointityyliä, olisi ollut parempi käyttää muuttumattomia tietorakenteita (immutable).

Tiedostot

Ohjelma käyttää tekstitiedostoja ja korttien kuvat sisältävää kuvatiedostoa. Ohjelma paloittelee kuvatiedostosta kunkin kortin kuvan omaksi kuvakseen. Käsiteltävät tekstitiedostot ovat pelin tallennustiedostoja. Ohjelma osaa lukea sekä kirjoittaa niitä. Tiedostoformaatti koostuu tekstilohkoista, jotka alkavat ’#’-merkillä. Tunnisteen perässä voi olla pöydällä olevat kortit, pakassa olevat kortit, yksittäisen pelaajan tiedot, tai vuorolaskurin luku. Pelaajan tiedot koostuvat jälleen tekstilohkoista, jotka alkavat ’%’-merkillä. Tämän tunnisteen perässä voi olla pelaajan nimi, mökit, pisteet, tieto onko pelaaja jakaja tai tietokonevastustaja, pelaajan pinossa ja kädessä olevat kortit. Yksittäistä korttia kuvaa merkkijono ”a,b,c”, jossa a on kortin arvo pöydällä, b kortin arvo kädessä ja c kortin maa, missä 1 on risti, 2 pata, 3 hertta ja 4 ruutu.

Testaus

Ohjelmaa testattiin lähinnä ajamalla sitä siinä vaiheessa, kun käyttöliittymä alkoi valmistua. Jos huomattiin virheitä ajaessa, niitä jäljitettiin lähinnä println-komennoilla, minkä avulla kaikki huomatut virheet saatiin helposti korjattua. Testaus toteutettiin pitkälti suunnitelman mukaisesti ajamalla sitä. Korttientarkistus-algoritmille tehtiin muutamia omia yksikkötestejä eri käyttötapauksilla ja ne läpäisivät testit. Testit ovat lähdekoodin mukana. Lisäksi tehtiin yksikkötesti, jolla varmistettiin, että tietokonevastustaja tekee siirron kutsuttaessa. Ohjelma läpäisee kaikki tehdyt testit.

Ohjelman tunnetut puutteet ja viat

Jos tietokonevastustajalle ja tarkistus-algoritmille annettaisiin paljon kortteja mistä valita siirtoja, ne saattaisivat olla hitaita, mutta pelitilanteessa tuskin tulee tilannetta, jossa tätä huomaisi. Dokumentin kirjoitushetkellä ei ole tiedossa olevia normaalin ajon aikaisia puutteita tai vikoja, mutta niitä luultavasti on olemassa. Tiedostoformaatissa ei olla otettu huomioon juurikaan virheentarkistusta, jos tallennustiedosto olisi käsin kirjoitettu. Kuitenkin ohjelma on toteutettu siten, että se käyttää aina samaa tallennustiedostoa, eli tallennettuna voi olla kerralla vain yksi

pelitilanne. Jos käyttäjä muokkaisi käsin tätä, voitaisiin saada virheitä esiintymään ohjelmaa ajettaessa.

3 parasta ja 3 heikointa kohtaa

Pelin tallennus ja uuden pelin alustus toimivat hyvin. Kuten edellä mainittu pelitilanteen lataus käsintehdystä tiedostosta on luultavasti pelin heikoin kohta, jos tiedostoa ei olla rakennettu kuten ohjelma sen rakentaisi. Lisäksi graafinen käyttöliittymä ei ole ainakaan visuaaliselta ilmeeltään ohjelman vahvuus. Tiedostonlukua voisi korjata lisäämällä virheenkäsittelyä enemmän. Graafisesta käyttöliittymästä taas olisi saanut paremman käyttämällä jotain muuta kirjastoa kuin Swingiä sen luomiseen.

Poikkeamat suunnitelmasta

Ohjelman lopputulos noudattaa pitkälti suunnitelmaa. Poikkeamat suunnitelmasta ovat Kortti-luokan korttientarkistus-metodit sekä jotkut metodit ovat toteutettu eri luokissa kuin suunniteltu, koska tämä koettiin helpommaksi/tehokkaamaksi toteutusvaiheessa. Ajankäytössä graafinen käyttöliittymä vei selkeästi eniten aikaa, kuten suunnitelmassa jo arveltiin. Tietokonevastustajia ei toteutettu enempää kuin yksi, vaikka oltiin suunniteltu toteutettavan ainakin kaksi erilaista. Tämän syynä, ettei aikaa ollut tarpeeksi ja useiden strategioiden toteuttaminen olisi tuntunut kuin olisi tehnyt lähes samaa koodia useaan kertaan. Toteutusjärjestys oli tismalleen sama kuin suunnitelmassa.

Toteutunut aikajärjestys ja aikataulu

Projektissa toteutettiin ensin luokat Pelikierros ja Kortti, sitten Pelaaja ja Tietokonevastustaja, joiden jälkeen Peli-objekti ja viimeisenä Käyttöliittymä. Käsitys joistain metodeista, kuten kortintarkistus muuttui useita kertoja toteutuksen aikana, jolloin niitä muutettiin vastaamaan käsitystä. Käyttöliittymän toteutus ja virheiden etsiminen kulkivat käsi kädessä ohjelman toteutuksen aikana, toisin kuin suunnitelmassa. Tarkkoja päivämääriä ei ole, mutta yksinkertaisimmat luokat kuten Pelikierros, Kortti ja Pelaaja toteutettiin maaliskuussa ja loput luokat, testaukset ja korjaukset huhtikuussa.

Arvio lopputuloksesta

Muut kurssit veivät paljon enemmän aikaa kuin olin suunnitellut, mikä söi useita kertoja projektin tekemisen jatkamista. Projekti saatiin kuitenkin valmiiksi juuri ennen määräaikaa ja olen tyytyväinen sen laatuun. Ohjelman testausta olisi voinut suunnitella paremmin ja enemmän, koska on kuitenkin epätodennäköistä, että se toimisi täydellisesti eikä siinä olisi mitään olemassa olevia virheitä. Ohjelman suurimpana puutteena on luultavasti tiedostosta lataus, mutta se toimii ainakin niin kauan tarkoituksenmukaisesti, kun käyttäjä ei muokkaa tallennustiedostoa tallennusformaatin vastaiseksi. Käyttöliittymän olisi voinut toteuttaa käyttäen Processingia eikä Swingiä, jolloin siitä olisi saattanut tulla parempi. Mielestäni toteutettu luokkajako ja käytetyt tietorakenteet ovat hyvin toimivia. Korttientarkistus-algoritmin olisi luultavasti voinut toteuttaa paremmin ja tehokkaammin. Olen kuitenkin tyytyväinen omaan työpanokseeni ja aikaansaannokseeni ja sitä mieltä, että ohjelma toimii melko hyvin. Ohjelmakoodia ei ole niin paljoa ja ohjelma on pilkottu moniin metodeihin, joten ohjelman rakenne sallii muutosten tai laajennusten tekemisen.

Viitteet

Materiaalina käytettiin Scalan perusluokkakirjastojen API-kuvausta ja kurssin materiaaleja.