Antti Mäkimattila ja Joona Kärkkäinen  
Shakkipeli

Turun Yliopisto

olio-ohjelmoinnin perusteet

Ohjaaja: Erno Lokkila

9.3.2020

[antti.j.makimattila@gmail.com](mailto:antti.j.makimattila@gmail.com)

[joona.karkkainen@gmail.com](mailto:joona.karkkainen@gmail.com)

Sisällys

Kansilehti [1](#_Toc34667266)

Sisällysluettelo [2](#_Toc34667267)

[Tehtävän kuvaus ja analysointi 3](#_Toc34667268)

[Ratkaisuperiaate 7](#_Toc34667271)

[Ohjelman ja sen osien kuvaaminen 8](#_Toc34667272)

[Testausjärjestely 9](#_Toc34667273)

[Liitteet 10](#_Toc34667274)

Tehtävänä oli toteuttaa 2-3 hengen ryhmissä Java-ohjelmointikielellä laaja ohjelmakokonaisuus harjoitustyönä, jonka aiheen pystyi valitsemaan annetuista aiheista, tai ehdottaa omaa aihetta.

Tehtävänantoon kuului, että ohjelmassa pitää olla vähintään 3 omaa luokkaa, joista tehdään olioita, käyttää hyväksi perintää tai rajapintaa, hyödyntää polymorfismia ja joko tallentaa ohjelman tila tietokantaan/tiedostoon tai tehdä graafinen käyttöliittymä ohjelmalle.

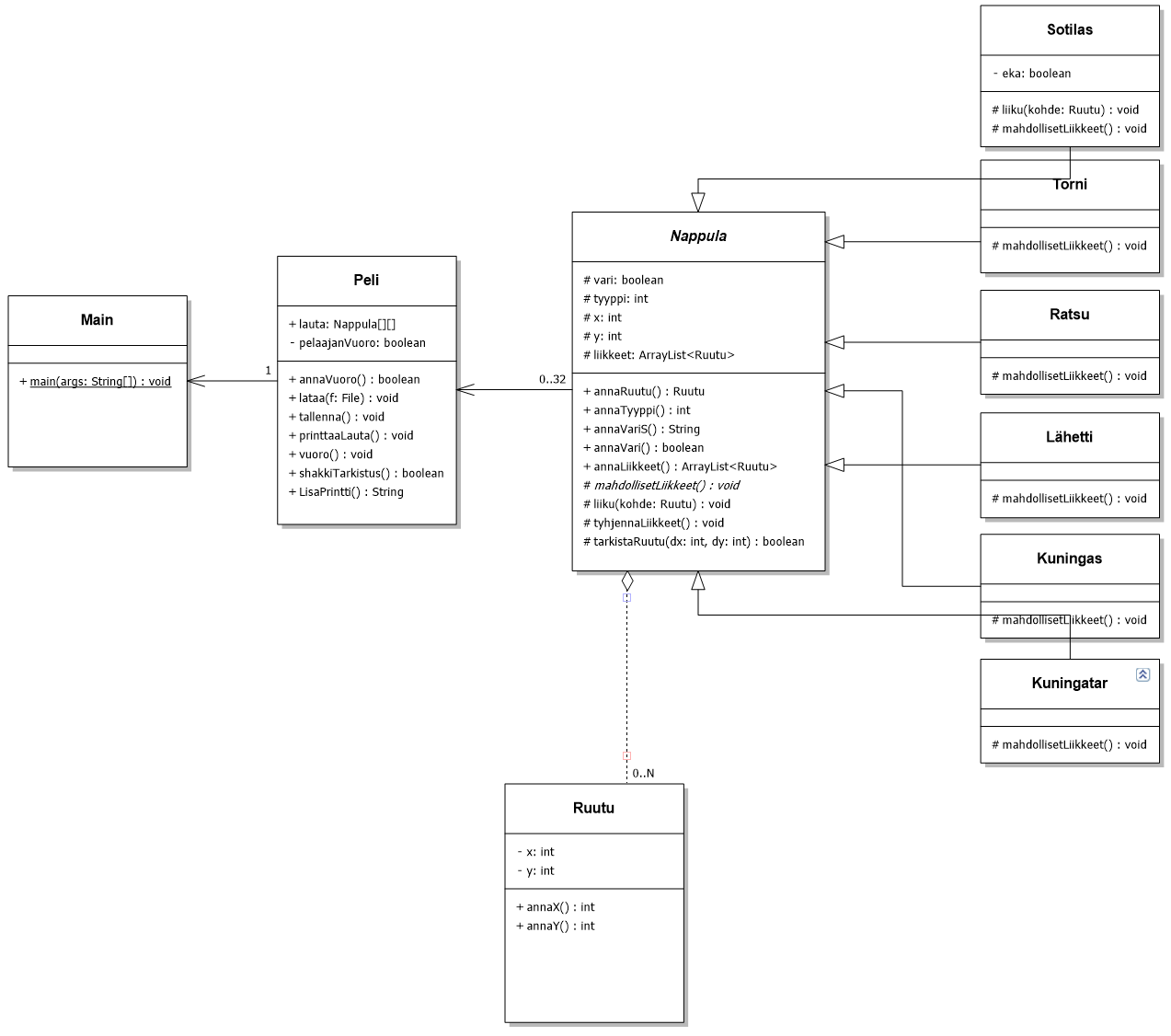
Päätimme tehdä tarjotuista vaihtoehdoista shakkipelin, ja vaikka aluksi suunnitelmassa oli graafisen käyttöliittymän lisääminen, päädyimme lopulta pelkkään ascii-grafiikkaan komentorivillä.

Kun ohjelman käynnistää, se kysyy, aloitetaanko uusi peli vai jatketaanko edellisestä. Jos pelaaja valitsee uuden pelin, ohjelma lukee default.txt -tiedostosta shakin aloitusasetelman ja laittaa valkoisen aloittamaan pelin. Jos taas pelaaja valitsee jatka, niin peli jatkuu viimeisestä tallennuksesta, joka löytyy save.txt tiedostosta. Vuoro menee sille, jonka vuoroon peli jäi viime kerralla.

Shakin 8x8 pelilauta saatiin helposti toteutettua kaksiulotteisella oliotaulukolla, jossa jokainen alkio sisältää joko Nappula-olion tai on tyhjä. Jokainen erillinen nappulatyyppi (sotilas, ratsu ym.) on toteutettu perimällä abstrakti luokka Nappula, josta löytyy suurin osa tarvittavista metodeista. Jokaisessa Nappulan perivässä luokassa on metodi, jolla selvitetään kyseisen Nappulan kaikki mahdolliset siirrot nykyisellä vuorolla.

Pelaajat valitsevat aina vuorollaan laudalta nappulan, jota haluavat siirtää. Tällöin heille annetaan lista kaikista mahdollisista siirroista, joita valittu nappula voi tällä vuorolla tehdä. Käyttäjä voi joko valita minkä siirron hän valitulla nappulallaan tekee, tai vaihtoehtoisesti vaihtaa vielä liikutettavaa nappulaa. Kun käyttäjä on valinnut jonkin siirron numerosyötteellä, siirretään valittu nappula pelaajan valitsemalle paikalle matriisissa. Jos tässä ruudussa on vastustajan nappula, se syödään, jolloin se katoaa laudalta. Vuoron loputtua ohjelma vielä tallentaa tekstitiedostoon nykyisen pelitilan, josta sen voi sitten ensi käynnistyksellä halutessaan ladata.

Ohjelma osaa ratkaista milloin kyseessä on shakki, mutta ei osaa selvittää pääseekö siirrolla pois shakkitilanteesta. Kun tilanne on shakissa, pelaajalle tulee mahdollisuus luovuttaa peli, jolloin ohjelma toteaa vastustajan voittajaksi ja ohjelma päättyy tähän.



Main

Shakin käynnistyessä kutsutaan luokkaa Main, jossa käyttäjä määrittelee, aloittaako Peli-olio uuden pelin vai jatkaako se edellisestä. Tämän jälkeen Main-luokassa oleva Peli-olio on vastuussa ohjelman toiminnasta niin kauan, kunnes peli loppuu. Pelin loputtua Main kertoo vielä käyttäjälle, kumpi pelaajista voitti pelin.

Peli

Peli-luokka sisältää suurimman osan pelin oikeasta toiminnallisuudesta, ja vastaa pelilaudan tilan luomisesta, muokkaamisesta, tulostamisesta, lataamisesta ja tallentamisesta. Peli-luokan ainoita attribuutteja ovat lauta ja pelaajanVuoro, joita kaikki luokan metodit käyttävät. Nimensä mukaan lauta säilyttää pelilaudan tilaa, kun taas pelaajanVuoro pitää sisällään tiedon siitä, kumman pelaajan vuoro on. Peli-luokka sisältää metodit laudan tilan tulostamiseen käyttäjälle, jonka avulla pelaaja seuraa pelin kulkua. Pelilaudan tila tulostetaan jokaisen vuoron alussa, jolloin pelaajan tulee valita jokin oma nappula laudalta. Pelaaja valitsee laudalta siirrettävän nappulan syöttämällä merkkijonon ohjelmalle, jonka jälkeen hän valitsee paikan, johon tätä nappulaa ollaan siirtämässä. Tämän jälkeen vuoro vaihtuu, ja peli tallennetaan. Vuoro vaihtuu niin kauan pelaajien välillä, kunnes toinen pelaajista voittaa. Pelin tallennuksessa muokataan lauta-matriisin tila tekstimuotoon ja tallennetaan se tiedostoon save.txt, latauksessa tiedostosta luetaan teksti ja muokataan se takaisin matriisiksi.

Nappula

Pelilaudan kaikki nappulat perivät abstraktin luokan Nappula. Nappula-luokka sisältää kaiken tarvittavan tiedon, mitä yksittäisestä nappulasta tarvitaan, kuten sen värin, tyypin, koordinaatit laudalla ja listan liikkeistä, joita Nappula voi vuorollaan tehdä. Edellä mainitut asiat ovat kaikki attribuutteja luokassa, joille kaikille on oma havannointimetodinsa. Väriä varten on olemassa kaksi havannointimetodia, joista toinen palauttaa värin boolean-arvona ja toinen merkkijonona ’V’ (valkoinen) tai ’M’ (musta). Nappula sisältää yhden abstraktin asetusmetodin mahdollisetLiikkeet(), jolla listan liikkeet tila asetetaan oikeaksi. mahdollisetLiikkeet() on määritelty erikseen kaikissa luokan Nappula perivissä luokissa, jonka lisäksi luokassa Sotilas on määritelty myös se, onko nykyinen siirto sen ensimmäinen. Muut luokan Nappula perivät luokat (Torni, Ratsu, Lähetti, Kuningas, Kuningatar) toteuttavat vain ja ainoastaan metodin mahdollisetLiikkeet(). Nappula-luokassa on vielä yksi metodi, tarkistaRuutu(int dx, int dy), joka kutsuttaessa kertoo, voiko nappula liikkua määriteltyyn ruutuun laudalla.

Ruutu

Ruutu-luokan ainoa tarkoitus on säilyttää kahta kokonaislukua x ja y, jotka ovatkin luokan attribuutteja. Luokka sisältää vain havannointimetodit annaX() ja annaY(), jotka nimensä mukaan palauttavat kokonaislukuna jommankumman attribuuteista.