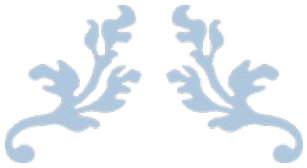
13.32016



Turun Yliopisto

Olio-ohjelmoinnin perusteet (TKO_2005) Harjoitustyö

Tero Yrjölä 507615 tekryr@utu.fi
Jussi Wallin 509674 juwawa@utu.fi
Niklas Niemelä 512613 niosni@utu.fi

Tehtävänanto:

Shakkipeli

Harjoitustyössä tulee mallintaa shakkilauta ja pelin eri nappulat. Tekoälypelaajaa ei tarvitse toteuttaa, mutta työn tulisi mahdollistaa ihmispelaajien väliset ottelut. Graafisesti pelipöydän ja pelin tilan voi esittää komentorivillä ascii-grafiikalla tai graafisella käyttöliittymällä. Toteutuksessa pitää myös huomioida eri pelinappuloiden sallitut siirrot ja hoitaa automaattisesti nappuloiden syönnit. Pelin logiikkaa ei muuten tarvitse toteuttaa, eli esimerkiksi pelin voittajaa ei tarvitse selvittää. Lisäksi pelin tila tulee olla mahdollista tallentaa, jotta peliä voi tarvittaessa jatkaa myöhemmin.

Päätimme valita harjoitustyöksemme shakkipelin, sillä se vaikutti sopivan haastavalta ja sisälsi mielenkiintoisia elementtejä. Ohjelmamme käynnistyy .bat tiedostosta, joka avaa konsolin, mihin piirtyy shakkilauta ansi-koodausta hyväksikäyttäen. Nappuloiden vihreä väri tarkoittaa valkoisen pelaajan nappuloita ja punainen mustia. Ohjelma ilmoittaa aina pelaajan vuoron joka siirron alussa. Nappulat on nimetty universaalien englanninkielisten lyhenteiden mukaan.

Nappuloita liikutetaan *move*-komentoa käyttäen, jonka jälkeen syötetään ensin liikutettavan nappulan koordinaatit [A-H][1-8] koordinaatistossa. Tämän jälkeen syötetään kohdekoordinaatti ja mikäli siirto oli laillinen, nappula siirtyy jälkimmäiseen koordinaattiin ja pelivuoro vaihtuu. Mikäli siirto ei ole hyväksyttävä, pelilauta piirretään uudestaan ja pelivuoro ei vaihdu. Komento *save* tallentaa pelin tilan tekstitiedostoon ja ilmoittaa, onko tallennus onnistunut. Komennolla *load* voidaan ladata tallennettu pelitila tekstitiedostosta. Komento *newgame* aloittaa uuden pelin. Viimeinen komento on *help*, mikä antaa tietoja väreistä ja mahdollisista komennoista.

Emme saaneet shakkimatti-tilannetta toimimaan kunnolla, joten tällä hetkellä shakki-tilanne on hieman arvaamaton.

Ratkaisuperiaate

Pelin alustaminen tapahtuu niin, että ohjelma luo uuden *ChessBoard*-olion, mikä perii luokan Serializable, *newGame()*-metodilla. *ChessBoard*-oliolla on attribuutit pelaavasta väristä ja pelilaudan koosta (8x8). Metodi *newGame()* alustaa *ChessBoard* olion lisäämällä pelinappulat oikeille paikoilleen. Luokalla *ChessBoard* on metodi *showBoard()*, mikä piirtää pelilaudan ja nappulat käymällä läpi koko koordinaatiston ja piirtäen nappulat oikeille paikoilleen laudalla käyttäen *case*-lausetta. Lisäksi *ChessBoard*-luokalla on metodi *isCheck*, mikä tarkastaa pelitilanteesta shakin. Metodi *move* vaatii lähtökoordinaatit ja maalikoordinaatit ja siirtää nappulan lähdöstä maaliin riippuen siitä, onko siirto mahdollinen. Metodi käyttää toista metodia *interpretCoord* hyödykseen siten, että se muuntaa käyttäjän syöttämät shakkilaudan koordinaatit (A-H, 1-8) ohjelman koordinaateiksi (x-suunnassa 0-7 ja y-suunnassa 0-7).

Toteutimme pelinappulat siten, että jokainen pelinappula perii abstraktin luokan *piece*, millä on attribuutteina väri (*enum PieceColour*), sijainti koordinaatistossa (*int x* ja *int y*) ja boolean attribuutit *hasMoved*, *isTaken* ja *canJump.* Jokaisella pelinappulalla on oma metodi *isLegalMove*, mikä tarkastaa, onko siirto laillinen. Esimerkiksi rook(torni)-nappulan metodi tarkastaa, onko kohderuutu tyhjä, liikkuuko nappula vain horisontaalisesti tai vertikaalisesti ja onko välissä muita pelinappuloita (toteutettu silmukalla käymällä läpi jokainen välissä oleva ruutu).

Moduulien kuvaus

Useat metodien nimet ovat itseään kuvaavia, joten oletamme, ettei niitä tarkemmin tarvitse selittää.

ChessBoard.java:

Pelilaudalla on metodit *addPiece(Piece, int, int), getBoard()* palauttaa pelilaudan, *getPiece(int, int), getId(Piece), isCheck(), isWhitePlayerTurn(), setWhitePlayerTurn(boolean), move(int, int, int, int)* ja *showBoard().*

Piece.java:

Jokainen pelinappula perii metodit *canJump(), getPlayer(), getX(), getY(), hasMoved(), isTaken(), move(ChessBoard, int, int), setHasMoved(Boolean), setTaken(Boolean), setX(int)* ja *setY(int).*

Lisäksi jokaisella nappulalla on oma @Override-metodi *(ChessBoard, int, int)*.

NewGame.java:

newGame(ChessBoard)

PieceColour.java:

Enum PieceColour.BLACK ja *PieceColour.White*

Start.java:

public static void main(String[] args)

interpretCoord(String ChessCoord), mikä muuttaa syötteen *move*-metodille sopivaksi.

executeCommand(Scanner), mikä lukee käyttäjän syötteen ja suorittaa toiminnot *move, help, save, load* ja *newGame.*

Testausjärjestely

Ohjelmaa testattiin siten, että kun piirtämismetodi oli saatu valmiiksi, niin nappuloiden siirtelyä koitettiin eri tavoin.

Ohjelman käyttöohje

Ohjelma käynnistyy ”GameStart.bat” komentojonotiedostosta, mikä avaa komentorivin ja peli alkaa. Kirjoittamalla komentoriville ”help” ja painamalla enter-painiketta, saat ohjeet näkyviin.

Vastuualueet

Jussi Wallin:

-liikkuvien nappuloiden logiikka

-pelin tallennus ja lataus

-Testaus

Tero Yrjölä:

-Grafiikan toteutus

-Dokumentointi

-Testaus

Niklas Niemelä:

-Uuden pelin aloitus

-Testaus