Toteutusdokumentti, Kalaha

Ohjelma koostuu neljästä luokasta:

Main:
Ohjelma suoritetaan tässä luokassa. Luo Board- ja Game-oliot, jotka sisältävät pelin säännöt ja kulun tarvittavine metodeineen ja muuttujineen.

Game:
Pelin kulku menee tämän luokan pelaa()-metodin mukaisesti. Tarvitsee MiniMax-olion minimax-algoritmia tietokoneen optimaalisen siirron selvittämiseen.

MiniMax:
Sisältää minimax-algoritmin, joka saa syötteenä pelaajan (1 tai 2), jonka paras mahdollinen siirto halutaan selvittää, pelilaudan, josta siirto lasketaan, tutkittavan syvyyden, joka määritetään metodin sisällä, sekä parametrit alfa ja beta, joiden avulla karsitaan turhia pelipuun oksia. Palauttaa 6-paikkaisen listan, jonka indeksien 0...5 arvot edustaa kyseisen indeksin siirrolla saavutettavaa pistemäärää. Algoritmi olettaa, että molemmat pelaavat optimaalisesti.

Board:
Tämä luokka huolehtii pelilaudan alustuksesta, tulostuksesta ja päivittämisestä tehtävän siirron mukaan, sääntöjä noudattaen.

Jokaisella luokan Board ilmentymällä on luokkamuuttujana Board olio “temp”, jota käytetään simuloivan siirron yhteydessä, jotta varsinainen pelilauta ei muuttuisi sen takia. Simuloiva siirto päivittää siis vain tämän temp-laudan pelitilannetta sekä uusiVuoro-muuttujaa, joka kertoo, saako siirrolla uuden vuoron.

Luokassa on myös tarkistus sille, onko peli ohi, eli onko jomman kumman pelaajan pelikupit tyhjänä.

(Lisäksi dokumenteissa Luokkakaavio.)

Aika- ja tilavaativuus:

Pelipuut ovat yleensä aikavaativuudeltaan suuria ja vain pienille, kevyille peleille suoritus onnistuu kohtuullisessa ajassa. Pelkällä minimax-algoritmilla kuusihaaraisen puun läpikäymisen aikavaativuus olisi luokkaa O(b^d), missä b on haarautumisaste, joka Kalahassa on 6 ja d on puun syvyys, joka Kalahassa on arviolta keskimäärin 20-30 (perustuu omaan testailuun).

Alfa-Beta -karsinta, jota tässäkin työssä olen käyttänyt, voi pienentää aikavaativuutta parhaimmillaan luokkaan O(b^(d/2)), eli puun syvyys puolittuu. Pahimmassa tapauksessa, jossa pienimmät arvot tutkitaan ensin, on aikavaativuus sama kuin minimax:lla ilman alfa-beta -karsintaa. Olen tässä työssä rajoittanut syvyyden 20, jotta ohjelma suorittaisi pelin kohtuuajassa. Ensimmäisen siirron etsimiseen menee tällä syvyydellä noin 4 sekuntia.

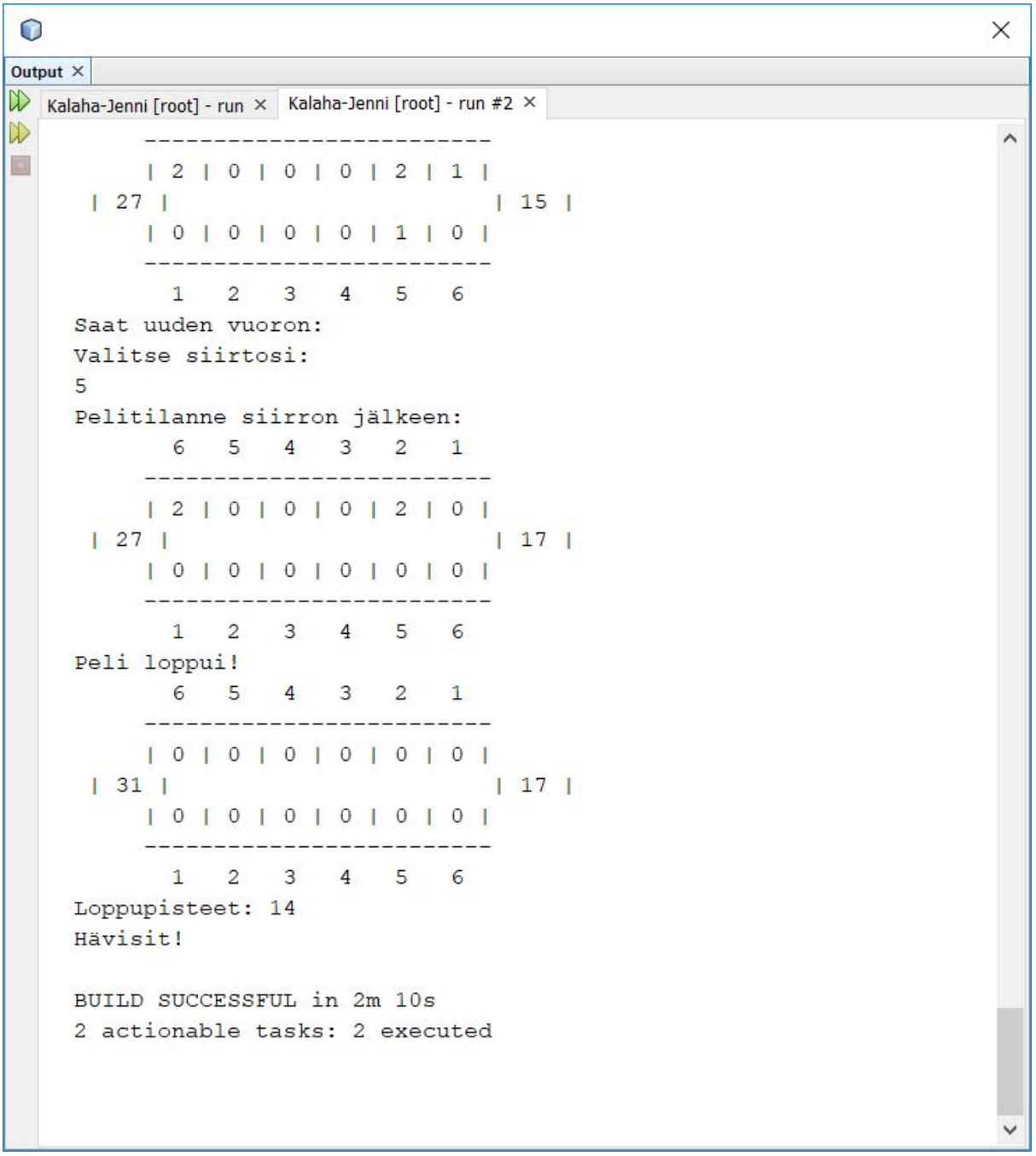
Tilavaativuus on minimax-algoritmilla O(bd) huolimatta siitä, onko alfa-beta -karsintaa käytetty vai ei.

Puutteet ja parannusehdotukset:

Pelissä on tarkistus sille, että käyttäjän antama syöte siirtoa tehtäessä kelpaa pelitilanteessa, mutta aloittajan arvonnassa jos pelaaja ei valitse 1 tai 0, tietokone aloittaa. Tähän olisi voinut lisätä tarkistuksen ja pyytää pelaajaa antamaan kelvollinen syöte (1 tai 0).

Jokin minimax-metodissa ei toimi halutulla tavalla. En keksinyt onko vika itse minimax:ssa vai alpha-beta -karsinnasta, joka on metodin sisällä. Kone pelaa kuitenkin ihan kohtuullisen hyvin, mutta se on mahdollista voittaa, mikä ei tietenkään ollut tässä tarkoitus.

Kuitenkin silloin tällöin jouduin tyytymään tappioon konetta vastaan:


Output
Kalaha-Jenni [root] - run
Kalaha-Jenni [root] - run #2
2	0	0	0	2	1
27		15
0	0	0	0	1	0
1 2 3 4 5 6
Saat uuden vuoron:
Valitse siirtosi:
5
Pelitilanne siirron jälkeen:
6 5 4 3 2 1
2	0	0	0	2	0
27		17
0	0	0	0	0	0
1 2 3 4 5 6
Peli loppui!
6 5 4 3 2 1
0	0	0	0	0	0
31		17
0	0	0	0	0	0
1 2 3 4 5 6
Loppupisteet: 14
Hävisit!
BUILD SUCCESSFUL in 2m 10s
2 actionable tasks: 2 executed