Jussi Wallin, Niklas Niemelä, Tero Yrjölä

Harjoitustyö

AIHE:

A: shakkipeli

Tarkoituksena on luoda komentorivillä toimiva ASCII-grafiikalla toteutettu kahden pelaajan peli. Peliin ei tule tekoälyä, mutta tunnistaa shakin sekä tornituksen.

Shakkipeliin on tarkoitus käyttää komentorivin värejä, joilla erotellaan valkoiset ja mustat nappulat sekä ruudut toisistaan. Piirtofunktio vaihtelee ruutujen värejä mukaillen shakkilaudan mustia ja valkoisia ruutuja ja piirtää nappulat vastaavissa väreissä kirjainyhdistelminä. Lautaa piirrettäessä funktio käsittelee NULL-arvon sisältämät paikat matriisissa tyhjinä ruutuina. Siirrot toteutetaan kirjoittamalla ne komentoriville, jonka jälkeen pelaaja näkee päivitetyn ikkunan. Shakkilauta objekti tuntee ja luo kaikki nappulat nappulamatriisiin, ja nappuloiden liikuttaminen toteutetaan siirtämällä ne matriisin taulukon paikasta toiseen. Jokainen nappula tuntee oman sijaintinsa, joka tallennetaan kokonaislukutaulukko muuttujaan.

Shakkitilanne tarkastetaan käymällä molempien kuninkaiden kohdalla läpi kaikki vastustajien nappulat ja tarkistetaan onko niillä sallittuja siirtoja kuninkaan ruutuun. Shakkitilanteessa peli estää siirtämästä muita nappuloita kuin kuningasta. Kuningasta siirrettäessä varmistetaan, ettei kuningasta yritetä siirtää uhattuun ruutuun.

Kaikille nappulatyypeille tehdään ennen siirtoa sen laillisuuden tarkastus. Jos nappula ei voi hypätä muiden yli tarkastetaan, että onko lähöpisteen ja halutun määränpään välillä muita nappuloita. Lopulta tarkastetaan onko halutussa määränpäässä oma nappula. Jos kaikkien tarkastusten jälkeen siirto on sallittu, palautetaan TRUE arvo laillisuustarkastus funktiosta. Jos yksikin tarkastus estää siirron tekemisen, funktio palauttaa arvon FALSE.

Luodaan luokat:

```
class chessBoard {
        private piece[][] matriisi;
        public void move(int[] posOrig, int[] posMoveTo);
        public void drawBoard();
        public int isCheck(); //palauttaa -1, 0 tai 1


}
abstract class piece {
        public enum pieceColour {
                WHITE,
                BLACK
        }
        private pieceColour player;
        private int[] position;
        private boolean hasMoved;
        private boolean isTaken;
        private boolean canJump;
        abstract void move(int x, int y);
        abstract boolean isLegalMove(int x, int y);
}
class soldier { } class rook{ } jne.
```