Projekat Blockade – Izveštaj

Faza 1

Projekat se sastoji od 5 fajla:

* Blockade.py – sadrži kod za unos osnovnih podataka o igri (Izbor oznake, veličina tabele, broj zidova) kao i sama logika odigravanja igre. Ukoliko je uključen debug mode (debugMode = true), korisnik neće unositi podatke već će biti kreirana osnovna igra sa predefinisanim parametrima.

Funkcije:

* + requestInputForSign() – zahtevanje unosa validnog broja – 1 za X, 2 za O simbol (Izbor simbola igrača, drugim simbolom će igrati računar kasnije).
  + requestInputForTableSize() – zahtevanje unosa broja vrsta(n) I kolona(m) tabele, maksimalno vrsta 22, kolona 28 (po uslovu zadatka)
  + requestInputForWallsNumber() – zahtevanje unosa broja zidova (min 1, max 10).
* Field.py – Klasa Field predstavlja jedno polje u tabeli(matrici).

Funkcije:

* + getSymbol() – vraća simbol(string) u zavisnosti od tipa polja
  + isEmptyWallField() – provera da li je u pitanju slobodno polje za zid
  + isWall() – provera da li je u pitanju polje sa zidom
  + setWall(color) – postavljanje zida u zavisnosti od boje
  + areWallsCrossing() – provera da li se zidovi seku u zavisnosti od boje novog zida
  + isFieldForPlayer() – provera da li je u pitanju polje igrača (validno polje za pijuna) – parna vrsta I kolona – više o tome u objašnjenju Table.py fajla
  + changeType(type) – promena tipa na type
* Enums.py – Fajl Enums sadrži sve enumeracije koje kod čine preglednijim i razumnijim.

Enumeracija FieldType se odnosi na tipove polja:

* + EMPTY – prazno polje,
  + X – polje sa X pijunom,
  + O – polje sa O pijunom,
  + VERTICAL\_WALL\_EMPTY/HORIZONTAL\_WALL\_EMPTY – slobodno polje za postavljanje vertikalnog/horizontalnog zida
  + VERTICAL\_WALL\_FULL /HORIZONTAL\_WALL\_FULL – polje koje sadrži vertikalni/horizontalni zid

Enumeracija PlayStatus se odnosi na status(fazu) igre jednog igrača.

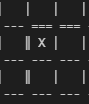
* Table.py – Klasa Table predstavlja tablu – matricu sa poljima. Elementi u parnim vrstama su prazno polje odnosno polje za igrača u slučaju parnih kolona ili vertikalni zid odnosno prazno mesto za vertikalni zid u slučaju neparnih kolona. Elementi u neparnim vrstama su horizontalni zidovi u slučaju parnih kolona ili prazna polja (nedostižne tačke) između zidova. Veličina matrice za obradu je 2\*n-1 vrsta i 2\*m-1 kolona (n broj vrsta table, m broj kolona table).

Funkcije:

* + initFields() – inicijalizuje matricu po navedenim pravilima
  + printTable() – štampa elemente tablice kreirajući vizuelno tablu po ugledu na tablu sa slajdova.
  + createPlayer() – kreira novog igrača, dodaje ga u niz i postavlja na tablu
  + **getFieldByRowAndColumn(row, column)** – vraća polje u matrici izračunavajući indekse matrice na osnovu unetog reda i kolone. Polje (1, 1) na tabli predstavlja polje (0, 0) u matrici koja se obrađuje. Vrsta matrice se računa po formuli: 2\*(row – 1), oduzimamo 1 zato što indeksi matrice počinju od nule dok indeks na tabli(vizuelno) počinje od 1. Zatim množimo sa 2 zato što pijuni mogu da se nalaze samo na parnim vrstama i kolonama. Kolona matrice se računa na sličan način, s tim što se umesto reda koristi prosleđena kolona table.
  + getFieldForWall() –
  + putWallOnPosition() – postavlja zid na unetu poziciju ukoliko je pozicija validna I ne sadrži druge zidove ili se kosi sa drugim zidovima.
  + isGameFinished() – proverava da li postoji pobednik u igri I ispusuje ga na standardnom izlazu.
  + **parseEnteredValueToTableIndex()** – provera da li je igrač uneo validan broj/slovo za vrstu/kolonu table. Unos je validan ukoliko je u pitanju broj ili slovo od A do Z (isključivo velika slova). Obzirom da vrsta I kolona počinju sa brojem 1, slovo A će predstavljati broj deset odnosno 10. vrstu/kolonu. Sva slova dobijaju vrednost u zavisnosti od svoje ASCII vrednosti. ASCII vrednost – 55. 55 oduzimamo zato što A ima vrednost 10, svako naredno slovo ima vrednost za jedan broj veći od prethodnog.
  + requestInputForPlayerPosition(sign) – zahteva unos vrste I kolone polja za poziciju izabranog pijuna od strane igrača sa znakom sign koji je na potezu
  + requestInputForWallPosition(sign) – zahteva unos boje, vrste I kolone polja za poziciju zida od strane igrača sa znakom sign koji je na potezu
* Player.py – Klasa predstavlja igrača (X ili O). Predstavlja svakog pijuna zasebno, biće ispravlljeno u sledećoj fazi. Sadrži atribut *type* koji predstavlja tip (X ili O), startingRow i startingColumn – početne pozicije pijuna koje su bitne radi provere kraja igre.

Funkcije:

* setPlayerOnTable() – postavlja pijuna na tabli.
* removePlayerFromCurrentPosition() – briše pijuna sa trenutne pozicije. Koristi se kod uspešnog pomeranja pijuna, kada treba da se obriše pijun sa prethodne pozicije.
* updatePlayerCordinates() – koristi funkcije removePlayerFromCurrentPosition() i setPlayerOnTable() kako bi se kroz jednu funkciju izvršio kompletan prelaz pijuna sa jednog polja na drugo.
* validateMoveForBoardDimensions(x, y) – vrši validaciju unetih kordinata x i y u odnosu na ivice table, da li je izabrana pozicija izvan granica table.
* validateMoveDirection(x, y) – vrši validaciju da li pravilno krecemo u pravcima gore, dole, levo, desno za jednu ili dve pozicije i da li se ispravno krecemo po dijagonali za jednu poziciju
* validateMoveForWalls(x, y) – vrši validaciju da li se zid nalazi na putu do odredišne pozicije. Prvi deo funkcije vrši validaciju za pravce: gore, dole, levo i desno. Drugi deo funkcije u odnosu u koji pravac ide ( poziva funkciju za gore levo - diagonalMoveUpLeft(x, y), itd. ).
* diagonalMoveUpLeft(x, y) – vrši proveru četiri različita slučajeva pozicije zidova.

* validateMoveForOtherPijuns(x, y) – vrši validaciju da li pijun može da se pomeri na polje koje je već zauzeo drugi pijun. Funkcija ignoriše slučaj kada se protivnički pijun nalazi na njegovoj startnoj poziciji.
* isWinner() – proverava da li je naš pijun došao do protivničke startne pozicije.
* play() –