**IZVEŠTAJ**

* **Algoritam korišćen za realizaciju kompjuterskog igrača:**

Nakon što igrač završi potez, prelazi se na potez kompjuterskog igrača. Poziva se metoda koja simulira bacanje kockica koja poziva metodu startTurn() (metoda koja simulira bacanje kockica poziva metodu startTurn() za trenutnog igrača, nije svesna koje je on vrste). Svaki igrač je realizovan kao zasebna nit. Ova metoda startTurn() budi nit komp igrača. Proces građenja stabla funkcioniše na sledeći način:

* Napravi se koreni čvor(tipa MAX) i njegova deca koja su svi mogući potezi koji se mogu napraviti iz trenutnog stanja(to su CHANCE ili TERMINAL čvorovi, u zavisnosti od dubine stabla).
* Ako je dubina stabla veća od jedan, sortiraju se čvorovi po rastućoj vrednosti statičke heuristike(heuristike izračunate samo za to stanje), uklanjaju se čvorovi sa istim stanjima(nema duplikata) i prelazi se na generisanje stabla
* Stablo se generiše rekurzivno tako što se za svaki CHANCE čvor pozove funkcija koja izgeneriše 21 čvor koji predstavlja moguće kombinacije kocki(CHANCE čvor će samo „prenositi“ vrednost mogućeg poteza iz svog roditeljskog čvora do svojih čvorova dece) i za svako od te dece se izgenerišu sva moguća deca(svi mogući potezi za kocke dodeljene tom čvoru). Ukoliko je dubina stabla veća od 2, uradi se odsecanje odredjenog broja čvorova na osnovu vrednosti statičke heuristike. Zatim se, za svako od prestale dece, ponovo poziva funkcija za generisanje, ali sa dubinom uvećanom za jedan.
* Kada se to završi, imamo izgenerisano stablo i onda se poziva funkcija expectimax() koja je zadužena da postavi vrednosti svim čvorovima u stablu.
* Ta funkcija radi na principu rekruzivnog postorder obilaska stabla, tako što se spušta do listova, računa heuristike, a onda propagira vrednosti na gore i to:
  + Ako je roditeljski čvor u koji smo se vratili MAX ili MIN on uzima heuristiku svog najboljeg deteta i njega bira kao izabrani čvor
  + Ako je roditeljski čvor CHANCE čvor on računa svoju vrednost kao zbir verovatnoća svakog deteta(za iste brojeve kocki ta verovatnoca je 1/36, za sve ostale je 1/18)
  + Kada se ovo završi, dobijamo potpuno izgenerisano stablo sa dodeljenim vrednostima, i onda sledeći potez pravimo tako što biramo onaj čvor koji ima istu vrednost heuristike kao naš čvor
* **Klase:**

**Main** – sadrži main metodu

**GameLogic** – predstavlja glavnu logiku igre, klasa zadužena za kontrolisanje rada igre

**public** GameLogic(Controler cont, **int** p); - konstruktor

**public** **void** setRedPlayer(Player pl);

**public** **void** setWhitePlayer(Player pl);

**public** **int** getWhiteBoard(); - broj belih žetona u igri

**public** **int** getWhiteBar(); - broj belih žetona na blotu

**public** **int** getRedBoard(); - broj crvenih žetona u igri

**public** **int** getRedBar(); - broj crevnih žetona na blotu

**public** **boolean** checkForMoves(**int** a, **int** b, **boolean** colorWhite) - proverava da li za zadate kocke postoji potez za trenutnog igraca

**private** **boolean** checkMoveFromBar(**int** x, **boolean** colorWhite) – proverava da li za zadatu kocku može da se pomeri žeton sa blota za trenutnog igrača

**public** ArrayList<**int**[]> generateMoves(**int**[] gameState, **int** num, **boolean** colorWhite, **int** numOnBoard) – generiše sve moguće poteze za zadato stanje igre, trenutnog igrača i zadatu kocku

**public** **int**[] redMoves(**int**[] gameState, **int** i, **int** num, **int** numOnBoard) – vraća novo stanje igre nakon pomeranja žetona i za vrednost kocke num za crvenog igrača

**public** **int**[] whiteMoves(**int**[] gameState, **int** i, **int** num, **int** numOnBoard) - vraća novo stanje igre nakon pomeranja žetona i za vrednost kocke num za belog igrača

**private int[]** setTemp(**int[]** gameState, **int** i, **boolean** colorWhite) – pomoćna metoda za prethodne 2

**public** **void** updateTable(**int**[] temp) – ažurira tablu(GUI)

**public** **void** startMatch() - postavlja početne parametre i započinje rundu

**public** **void** game() – kontroliše igru, postavlja potrebne parametre, utvrđuje da li je došlo do kraja runde/meča i poziva trenutnog igrača da započne potez

**public** **void** endOfMove() – završava trenutni potez i poziva game()

**private** **boolean** pat() – utvrđuje da li je došlo do pat pozicije

**public** **void** throwDices() – simulira bacanje kockica i poziva trenutnog igrača da napravi potez(ako može)

**public** **void** diceDet() – režim bacanaj 1 kocke kada se utvrđuje koji igrač igra prvi

**public** String whitePlayerName()

**public** String redPlayerName()

**public** Player getCurrentPlayer()

**public** **int** getdiceOne()

**public** **int** getdiceTwo()

**public** **void** secondMove() – podešava neophodne parametre kako bi ljudski igrač napravio potez za vrednost 2. kocke

**public** **void** setGameScreen(GameScreen game)

**public** **void** removeClicked()

**public** **int** currentPlayerBar()

**public** **int**[] getCurrentState()

**public** **int** getCurrentBar()

**public** **int** getCurrentOut()

**public** Node getTreeRoot()

**public** **int** getNextPlayerBar()

**public** **int** getNextPlayerOut()

**public** **void** disablePlayers()

**public** Color compPlayerColor()

**Bar** – GUI komponenta, izvodi se iz Jpanel – a, predstavlja „blot“

**public** Bar(); - konstruktor

**public** **void** addChecker(GUIChecker c) – dodaje zadati žeton na blot

**public** GUIChecker removeChecker(Color color) – uklanja žeton zadate boje sa blota

**protected** **void** paintComponent(Graphics g) – override istoimene metode roditeljske klase

**public** **void** empty() – resetovanje blota

**Branch** – Klasa nepohodna za vizuelizaciju stabla, predstavlja granu

**public** Branch(**int** one, **int** two); - konstruktor

**public** String toString(); – override metode toString

**Controler –** klasa zadužena za prelaze između panela, tj. Stadijuma igre

**public** Controler(WelcomeScreen ws); - konstruktor, dodeljuje mu se početni screen

**public** **void** twoPlayersGameSpec() – otvaranje prozora za izbor specifikacija za igru igrač protiv igrača

**public** **void** CompGameSpec()– otvaranje prozora za izbor specifikacija za igru igrač protiv kompjutera

**public** **void** start2playerGame(String p1, String p2, String points) – stvaranje nepohodnih elemenata za igru igrač protiv igrača

**public** **void** startCompGame(String username, String color, String points, String depth) stvaranje nepohodnih elemenata za igru igrač protiv kompjutera

**public** **void** resetMatch() – započinjanje nove runde

**public** **void** finishGame() – kraj igre, gašenje programa

**public** **void** startAllOver() – vraćanje na početni ekran

**GameScreen** – predstavlja „glavni ekran“, tj. Ekran u kome se igra, proiširuje Jframe.

**public** GameScreen(Controler cont, GameLogic gl); - konstruktor

**private** **void** addTreeButton(etf.backgammon.mm140457d.game.Color compColor);

**private** JButton getTreeButton();

**private** **void** treeButtonAction() – stvaranje grafičkog stabla, actionListener za treeButton

**private** Row getRow(**int** i, **int** dir);

**private** JPanel getGamePanel();

**public** OutBar getOutBar(); – dohvatanje trake za žetone koji su napustili igru

**public** Bar getBar(); – dohvatanje Blot trake

**private** JPanel getHelpPanel();

**private** JPanel getLabelPanel();

**public** JLabel getPlayingLabel()-dohvatanje labele na kojoj se ispisuje koji je igrač na redu

**public** JLabel getHelpLabel() – dohvatanje labele za pomoć

**private** JTextPane getTextPaneRed();

**private** JTextPane getTextPaneWhite();

**private** JButton getDiceButton() – dugme za bacanje kocki, u okviru njega postoji actionListener koji poziva odgovarajuću metodu za simuliranje bacanja kocki

**public** JLabel getDiceOne() - dohvatanje kocke 1

**public** JLabel getDiceTwo() – dohvatanje kocke 2

**private** JPanel getLeftPanel();

**private** JPanel getRightPanel();

**private** **void** setGame() – dodavanje žetona na tablu za početni raspored

**public** Row[] getRows();

**public** **void** setGameLogic(GameLogic gl);

**public** **void** disableButton();

**public** **void** disableDices();

**public** **void** disableCheckers();

**public** **void** enableButton();

**public** **void** enableDices();

**public** **void** enableCheckers();

**public** **void** turnDetermined(**boolean** b);

**public** **void** restart() – resetuje tablu za novu partiju

**private** **void** updateWhitePane() – ažurira informacije na textPane-u za belog igrače

**private** **void** updateRedPane()– ažurira informacije na textPane-u za belog igrače

**public** **void** showGraphicTree(Forest<Node, Branch> tree, Node root) – stvara deo grafičkog stabla(tj dodaje ga u model grafičkog stabla) tako što dodaje svu decu zadatog čvora root

**public** **void** TreeButtonEnabled(**boolean** b)

**GameSpecScreen –** apstraktna gui klasa, proširuje JFrame, predstavlja osnovu za klase PCGameSpecScreen i PPGameSpecScreen.

**public** GameSpecScreen(Controler cont); - konstruktor

**private** **void** makeRadioGruop();

**private** JLabel getLblUsername1();

**private** JTextField getTextFieldUsername();

**private** JLabel getLblPoints();

**private** JRadioButton getRadioButton1();

**private** JRadioButton getRadioButton3();

**private** JRadioButton getRadioButton5();

**private** JRadioButton getRadioButton7();

**private** JRadioButton getRadioButton11();

**private** JRadioButton getRadioButton15();

**protected** String getSelectedButtonText(ButtonGroup buttonGroup); – izdvaja tekst selektovanog radio button-a iz zadate grupe

**PPGameSpecScreen** – proširuje klasu GameSpecScreen, predstavlja ekran za izbor specifikacija za 2 igrača

**public** PPGameSpecScreen(Controler cont); - konstruktor

**private** JLabel getLblUsername2();

**private** JTextField getTextFieldUsername2();

**private** JButton getBtnStart();

**PCGameSpecScreen** – proširuje klasu GameSpecScreen, predstavlja ekran za izbor specifikacija za igrač protiv kompjutera

**public** PCGameSpecScreen(Controler cont); - konstruktor

**private** **void** setDepthRadioGroup();

**private** **void** setColorRadioGroup();

**private** JLabel getLblCheckerColor();

**private** JLabel getLblTreeDepth();

**private** JButton getBtnStart(); – sadrži actionListener za dugme start, poziva metodu kontrolera za započinjanje igre

**private** JRadioButton getRdbtnRed();

**private** JRadioButton getRdbtnWhite();

**private** JRadioButton getDepth2();

**private** JRadioButton getDepth3();

**GraphicTree** – proširuje Japplet, predstavlja vizuelizaciju stabla

**public** GraphicTree(Node root, GameScreen gameScreen); - konstruktor

**WelcomeScreen** – Početni ekran, otvara se prilikom pokretanja aplikacije

**public** WelcomeScreen(); - konstruktor

**private** JButton getPVSpButton() – sadrži actionListener, prilikom klika prelazi se na odgovarajući ekran za specifikacije

**private** JButton getPVScButton();– sadrži actionListener, prilikom klika prelazi se na odgovarajući ekran za specifikacije

**OutBar –** predsavlja GUI traku na kojoj se nalaze žetoni izbačeni iz igre, proširuje Jpanel

**public** OutBar(); - konstruktor

**public** **void** addChecker(GUIChecker c); - dodaje zadati žeton na traku

**public** **void** empty(); - resetuje na početno stanje, prazni traku

**Row –** proširuje Jpanel, predstavlja jedan „red“ na tabli, odnosno jedno moguće polje za smeštanje žetona(redova ukupno ima 24)

**public** Row(**int** dir, **int** rn); - konstruktor

**public** **void** addChecker(GUIChecker c, **int** pos); - dodaje zadati žeton u red

**public** GUIChecker removeChecker(); - uklanja žeton iz reda

**protected** **void** paintComponent(Graphics g); - override odgovarajuće roditeljske metode

**public** GUIChecker getTopChecker(); - dohvatanje žetona na vrhu

**public** **int** getCheckerNum();

**public** etf.backgammon.mm140457d.game.Color getCheckerColor();

**public** **void** setEnableCheckers(**boolean** b); - enable/disable svih žetona u redu

**public** **void** empty() – resetuje, tj. Prazni red

**GuiChecker –** proširuje Jpanel, predstavlja žeton za igru

**public** GUIChecker(**boolean** ot, **int** rn, etf.backgammon.mm140457d.game.Color color, **int** p, GameLogic gs); - konstruktor, implementira actionListener za žeton

**public** **boolean** getOnTop();

**public** **void** setOnTop(**boolean** ot);

**public** **int** getRowNum();

**public** **void** setRowNum(**int** v);

**public** etf.backgammon.mm140457d.game.Color getColor();

**public** **void** setClicked(**boolean** b);

**private** **void** clickedChecker(); - poziva odgovarajuće metode kada je kliknuto na žeton

**private** **void** callMakeAMove(JLabel checkingDice, **int** callingDice); - poziva trenutnog igrača da započne potez

**public** **void** setPosition(**int** pos);

**protected** **void** paintComponent(Graphics g); - override odgovarajuće roditeljske metode

**Node –** predstavlja jedan čvor stabla koji nosi određene informacije sa sobom

**public** Node(Type type, **int**[] state, **int** barEnemy, **int** barMe, **int** outEnemy, **int** outMe); - konstruktor

**public** Type getType();

**public** **void** setText(); - postavlja stanje kao tekst

**public** **void** hideText();

**public** **double** getValue();

**public** **void** setValue(**double** d);

**public** **void** addChild(Node child, **int** i); - dodaje dete na poziciju i

**public** ArrayList<Node> getChildren();

**public** **int**[] getFinalState();

**public** **int**[] getFirstState();

**public** **void** setFirstState(**int**[] state);

**public** **void** updateStatistic(**int**[] state, **boolean** colorWhite); - ažurira određena polja koja predstavljaju razliku trenutnog i prethodnog stanja

**public** **int** getHomeNum(**boolean** colorWhite); - broj žetona u home board-u

**public** **double** section2num(**boolean** colorWhite); – broj žetona u delu pored home board-a

**public** **boolean** sameAs(Node node); - poredi 2 čvora

**public** **double** getEnemyHomeNum(**boolean** colorWhite); - žetoni u protivnikovom home board-u, sa dodeljenim težinama

**public** **int** getDiceOne(); - prazno telo

**public** **int** getDiceTwo(); - prazno telo

**public** String toString();

**MinMaxNode** – proširuje klasu Node dodajući neke funkcionalnosti čvorovima tipa MIN i MAX

**public** MinMaxNode(Type type, **int** [] state, **int** d1, **int** d2, **int** barEnemy, **int** barMe, **int** outEnemy, **int** outMe); - konsturktor

**public** **int** getDiceOne(); - override metode

**public** **int** getDiceTwo(); - override metode

**public** **boolean** dicesEqual(); - poredi da li su kocke jednake

**public** **void** setChosen(Node c); – postavlja izabranog potomka

**public** Node getChosen();

**Player –** apstraktna klasa igrač

**public** Player(Color cl, String n, GameLogic gl); - konstruktor

**public** **abstract** **void** throwDice();

**public** **abstract** **void** startTurn(**boolean** second);

**public** **void** makeAMove(GUIChecker checker, **int** diceNum, **boolean** firstEntry); - disable nekih polja

**public** etf.backgammon.mm140457d.game.Color getColor();

**public** **abstract** **void** throwDices();

**public** **abstract** **void** run();

**public** **void** setGameScreen(GameScreen game);

**public** String name();

**public** **void** stopPlaying(); - postavlja flag za run na false

**HumanPlayer –** predstavlja ljudskog igrača

**public** HumanPlayer(Color cl, String name, GameLogic gl); - konstruktor

**public** **void** throwDices(); - ažurira GUI da naznači igraču da treba da baci kocke

**public** **void** startTurn(**boolean** second); - ažurira GUI da naznači igraču da treba da napravi potez

**public** **void** makeAMove(GUIChecker checker, **int** diceNum, **boolean** firstEntry); - postavi flag za run metodu na true(pokreće nit)

**public** **void** throwDice(); - ažurira GUI da naznači igraču da treba da baci kocku da se odredi ko igra prvi

**public** **void** run(); - poziva ciklički play()

**private** **void** play(); - glavna metoda zaduŽena da za izabrane vrednosti igrača proveri da li je dozvoljen potez, ako jeste ažurira GUI i omogući da se odigra drugi potez

**CompPlayer –** predstavlja kompjuterskog igrača

**public** CompPlayer(Color cl, String name, GameLogic gl, **int** td); - konstruktor

**public** **void** startTurn(**boolean** second); - postavlja flag za pokretanje niti

**private** **void** formPartOfTree(**int**[] currState, **int** dice1, **int** dice2, MinMaxNode parent, **int** depth, **boolean** colorWhite); - generiše poteze za prvu kocku i poziva metodu koja generiše za sledeću i stavlja u stablo

// generise stanja za 2. kocku, formira cvorove i ubacuje ih u stablo

**private** **void** stageTwoTreeFormation(**int**[] state, **int** dice1, **int** dice2, Type type, Node parent, **boolean** colorWhite, **int** depth, **boolean** barMoved) - generise stanja za 2. kocku, formira cvorove i ubacuje u stablo samo različite čvorove

**private** **void** generateTree(Node root, **int** depth); - rekurzivno generiše stablo

**private** **void** expectimax(**int** currentDepth, Node root) – rekurzivno određuje i dodeljuje težine(vrednosti) čvorovima

**private** **double** heuristic(Node root, **int** treeDepth) – izračunava heuristiku za zadati čvor na zadatoj dubini

**public** **void** throwDice(); - simulira bacanje 1 kocke za odredjivanje poteza

**public** **void** run(); - poziva play();

**public** **void** play(); - generiše koren i prvi nivo, a onda, ako je dubina stabla veća od 1, poziva metode se izgeneriše celo. Potom poziva metodu expectimax i na kraju bira izabrani potez da se odigra i poziva ažuriranje GUI-ja