**BACKGAMMON**

**Popescu Ioana-Maria**

**Grupa 231/2**

**Introducere:**

**Jocul de table contine 2 clase care instanteaza obiectele ce urmeaza a fi folosite in forma. Programul va afisa atat table de joc, cat si zarurile si piesele care sunt scoase din joc de catre oponent.**

**A picture containing knife, train

Description automatically generated**

**Obiective:**

* **Construirea tablei de joc si a pieselor;**
* **Implementarea functiilor de mutare a pieselor pe table de joc;**
* **Implementarea zarurilor;**
* **Implementarea inteligentei artificiale;**

**Definitii. Abrevieri. Acronime**

**In forma principala am definit o serie de variabile globale:**

* **Int DiceNumber = 0; -> variabila va fi folosita pentru identificarea zarului (=1 primul zar, =2 al 2-lea zar, =3 suma celor 2 zaruri).**
* **Int MiniScoreDice = 0; -> aceasta variabila va contine mereu numarul de mutari pe care le poate face player-ul in functie de zaruri.**
* **Int CoordX, CoordY; -> coordonatele punctelor x si y.**
* **Bool IsDouble; -> arata daca valorile zarurilor sunt egale.**
* **Int PositionPieceBackOnBoard = 0; -> arata pozitia in care revine piesa jucatorului “Human” pe table de joc.**
* **Int[,] MatrixBoard; -> reprezinta matricea in care se afla piesele de pe table (daca MatrixBoard[i,j] = 0 => pe acea pizitie nu se afla nicio piesa**

**MatrixBoard[i,j] = 1 => pe acea pozitie se afla o piesa neagra**

**MatrixBoard[i,j] = 2 => pe acea pozitie se afla o piesa rosie)**

* **Int[,] OutPiecesMatrix; -> reprezinta matricea in care se afla piesele scoase de oponent de pe table**

**(daca OutPiecesMatrix[i,j] = 0 => pe acea pizitie nu se afla nicio piesa**

**OutPiecesMatrix[i,j] = 1 => pe acea pozitie se afla o piesa neagra**

**OutPiecesMatrix[i,j] = 2 => pe acea pozitie se afla o piesa rosie)**

**Descrierea proiectului:**

**Proiectul contine, din punctul de vedere al continutului, 2 clase:**

* **Dice**
* **Player**

**Clasa Dice:**

**{**

**public int Cube1 { get; set; }**

**public int Cube2 { get; set; }**

**public bool IsDouble { get; set; }**

**private Random random = new Random();**

**public void RollDice()**

**{**

**Cube1 = random.Next(1, 7);**

**Cube2 = random.Next(1, 7);**

**if (Cube1 == Cube2) IsDouble = true;**

**else IsDouble = false;**

**}**

**}**

**Dupa cum se observa, aceasta clasa contine informatii specifice zarurilor.**

**Clasa Player:**

**{**

**public enum PlayerColor { Black, Red}**

**public string Name { get; private set; }**

**public PlayerColor playerColor { get; set; }**

**public bool isTurn { get; set; }**

**public Player(string name, PlayerColor color)**

**{**

**Name = name;**

**playerColor = color;**

**}**

**}**

**In forma principala se afla toate functiile implementate pentru realizarea logicii jocului.**

**Regions:**

1. **Initialization**
   1. **Functii:** 
      1. **InitializeComponentsOnBoard() -> acesta functie se ocupa cu initializarea pieselor pe table de joc.**
2. **Dice** 
   1. **Functii:**
      1. **RollTheDice\_click()**

* **La apasarea butonului “Roll”, aceasta functie apeleaza metoda RollDice din clasa Dice si potriveste, pentru fiecare zar, imaginea valorii lui.** 
  + 1. **Cube1Click()**
    2. **Cube2Click()**
    3. **SumCubesClick()**
* **In aceste 3 functii se verifica daca cele 2 zaruri au valori egale. In cazul afirmativ, DiceNumber va avea numarul zarului si MiniScoreDice se fi incrementat;**

1. **Game**
   1. **Functii:**
      1. **Redraw()**

* **Aceasta functie are rolul de a desena atat table de joc, cat si matricea in care sunt piesele scoase din joc.**
  + 1. **label3\_TextChanged()**
* **Rolul acesteia este de a bloca butonul de “Roll” pana cand se va schimba player-ul.**

1. **Checkers Control**
   1. **Functii:**
      1. **DeletePictureBox()**

* **Se ocupa cu stergerea piesei de pe table.**
  + 1. **DeletePictureBoxOut()**
* **Se ocupa cu stergerea piesei scoase de pe table.**

1. **Human Player Controls**
   1. **Functii:**
      1. **Pic\_OnClickBlack()**

* **In aceasta functie sunt implementate mutarile pe care le poate face player-ul “Human” pe table de joc**
  + 1. **GetOnBoardPos1()**
    2. **GetOnBoardPos2()**
    3. **GetOnBoardPos3()**
    4. **GetOnBoardPos4()**
    5. **GetOnBoardPos5()**
    6. **GetOnBoardPos6()**
* **Toate aceste functii modifica variabila PositionPieceBackOnBoard cu valoarea label-ului apasat.**
  + 1. **OutPics\_OnClickBlack()**
* **Aceasta functie preia coordonatele piesei din afara si apeleaza urmatoarea functie pentru a se insera pe table de joc.**
  + 1. **InsertChecker()**
* **Aceasta functie pune piesa scoasa de adversar inapoi pe table de joc.**

1. **AI Player Controls**
   1. **Functii:**
      1. **MoveAI\_Click()**

* **Acesta functie a fost implementata pentru ca player-ul “AI” sa mute piesele pe table de joc.**
  + 1. **InsertRedChecker()**
* **Aceasta functie pune piesa scoasa de adversar inapoi pe table de joc.**
  + 1. **RandomRedChecker()**
* **Aceasta functie allege random o piesa pe care player-ul “AI” sa o mute.**
  + 1. **GetAllPosibbleCheckersToMoveRedPlayer()**
* **Acesta functie a fost implementata pentru a se obtine toate piesele de pe table care se pot muta de catre player-ul “AI”. (pentru implementarea algoritmului ExpectiMiniMax).**

1. **Both Players Controls**
   1. **Functii:**
      1. **SwitchPlayer()**

* **Aceasta functie a fost implementata pentru a se realiza schimbarea player-ului.**
  + 1. **MoveThePieceToMatrix()**
* **Acesata functie muta piese de pe table de joc in matricea in care se afla piesele scoase de pe table de catre adversar.**