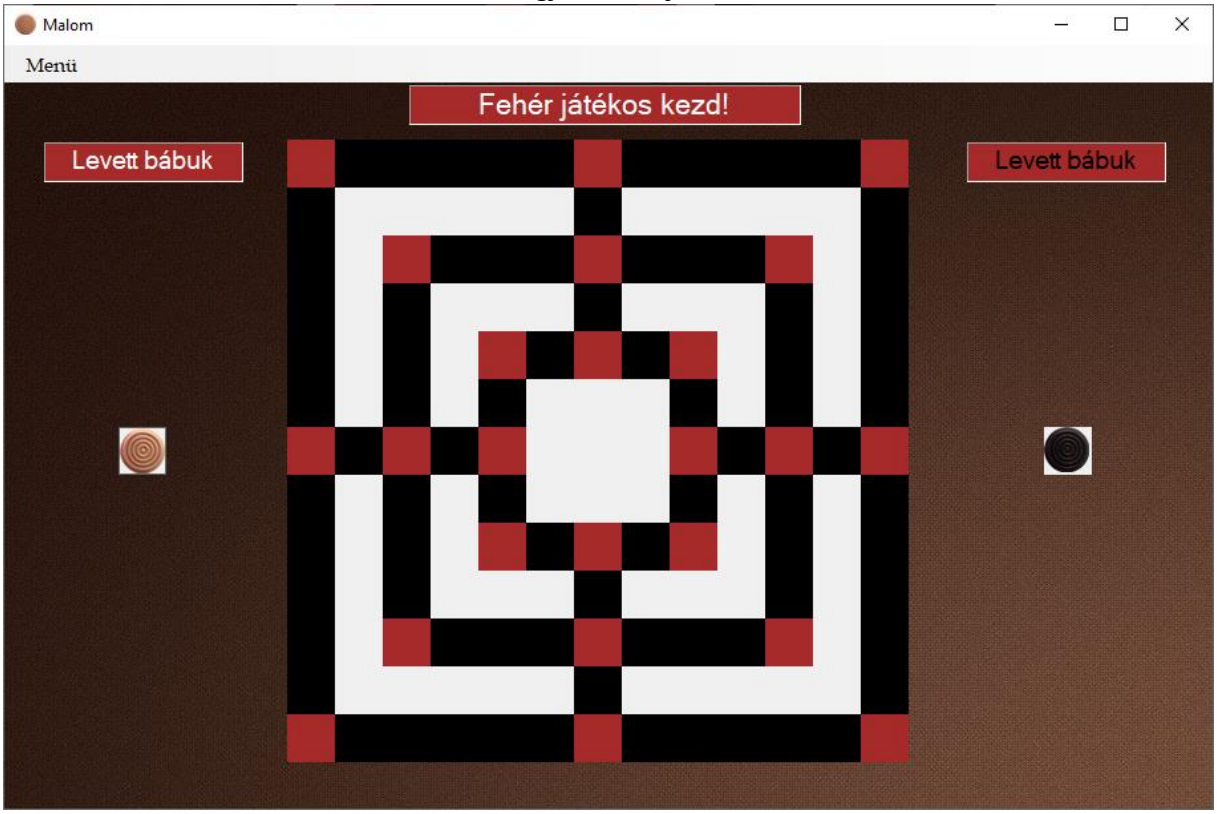
Tartalom

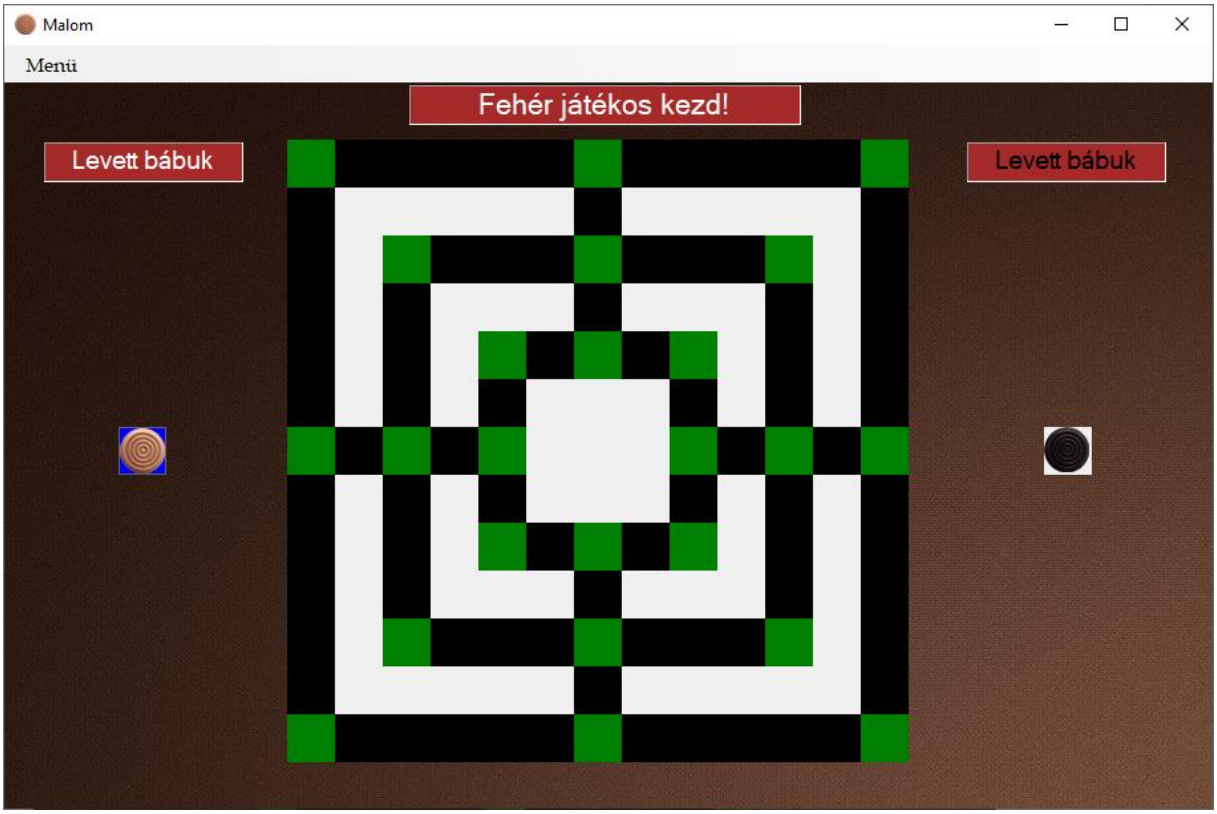
Felhasználói dokumentáció

A malom használata hasonló a sakkéhoz. A malom elindítása egy menüt dob fel, ahol ugyanúgy lehet új játékot létrehozni, illetve betölteni.

Miután kiválasztottuk mit szeretnénk, megjelenik a játék felhasználói felülete:



Itt is a fehér játékos kezd. A két kezdőbábu a tábla mellett találhatóak. Ezek a kezdőbábuk, addig lesznek ott, amíg a játékosok le nem rakják a 9-9 bábujukat a táblára. Ahhoz, hogy le lehessen rakni egy bábut csak rá kell nyomni. A megnyomás után a táblán azoknak a mezőknek lesz zöld a színe, ahova a játékos léphet. Értelemszerűen a barna mezők jelzik a játszható mezőket.



Itt is a kijelölt bábu mezője kék lesz. Ha egyik játékosnak sikerül malmot alkotnia, akkor le vehet egy ellenséges bábut. Ezt egész egyszerűen úgy teheti meg, hogy ráklikkel arra a bábura, amit le szeretne venni, viszont olyan bábut nem vehet le, ami malmot képez (kivéve, ha a pályán lévő ellenfél bábui mind malomban vannak). A kezdőbábukból (ha van még) nem tud egyik játékos sem elvenni. Miután mind a két játékos letette az összes kezdőbábuját, a táblán lévő bábukkal csak egyesével léphetnek vízszintesen és függőlegesen, viszont, ha 3 db bábuja marad a fehér vagy fekete játékosnak, akkor „ugrálhat”. Az a játékos veszít, amelyiknek csak két bábu marad a táblán vagy nincsen lépési lehetősége. Akármelyik játékos nyer, utána már nem léphetnek egyetlen bábuval sem.

Mind a két oldalon látható a levett bábuk felirat, ha egy játékos leveszi az ellenfele bábuját, akkor oda kerül.

A bal felső sarokban található egy menü, ahol tudunk új játékot létrehozni, menteni, betölteni, vagy kilépni. Mentésnél ügyelni kell arra, hogy a játékot malom helyzetben NEM lehet elmenteni. Későbbiekben ez ki lesz javítva.

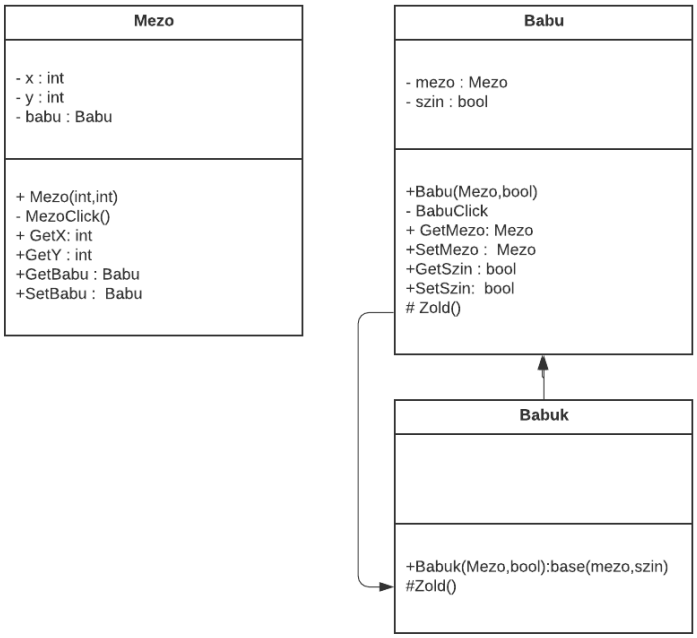
Fejlesztői dokumentáció

User Stories, User Cases, UML

User Stories:

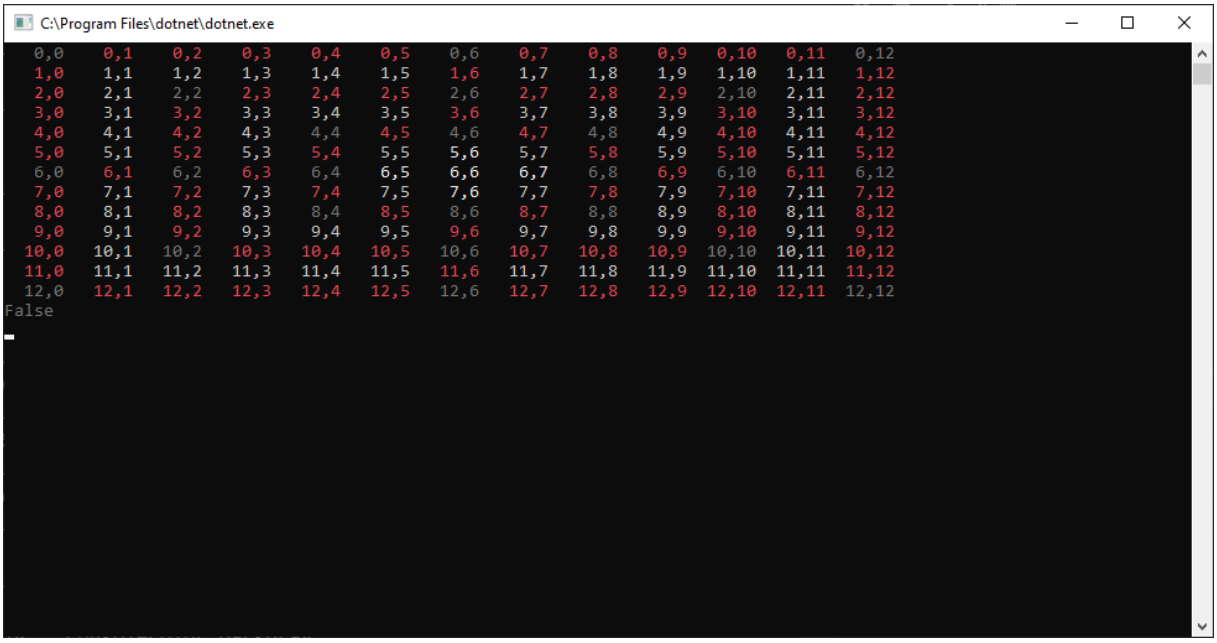
User Types	Epic	User Stories
Felhasználó	Menü kezelés	Felhasználó tud új játékot létrehozni
		Felhasználó tud játékot menteni
		Felhasználó tud játékot betölteni
		Felhasználó ki tud lépni a programból
Felhasználó	Bábuk mozgatása	Felhasználó tudja mozgatni a bábukat
		Felhasználó tud bábukat levenni

UML:



A tábla

A tábla létrehozására egy Mezo nevű osztályt hoztam létre, aminek van egy x és egy y koordinátája. Erről az osztályról később írok majd. A tábla létrehozásához létrehoztam egy Mezo típusú 13x13-as mátrixot, aminek neve tablakellek. Megnéztem melyek azok az indexek, ahol a táblának barnának kell lennie (játszható része a táblának). Segítségül konzol alkalmazásban hoztam létre egy ugyanilyen mátrixot és ki irattam az indexét.



Így könnyen látható volt, melyek azok az indexek, ahol színt kell változtatni.

```
for (int i = 0; i < tablakellek.GetLength(0); i++)
{
    for (int j = 0; j < tablakellek.GetLength(1); j++)
    {
        tablakellek[i, j] = new Mezo(i, j);
        tablakellek[i, j].SetBounds(j * 36, i * 36, 36, 36);
        //Nagy
        if (j == 0)
            tablakellek[i, j].BackColor = Color.Black;
        if (j == 12)
            tablakellek[i, j].BackColor = Color.Black;
        if (i == 0)
            tablakellek[i, j].BackColor = Color.Black;
        if (i == 12)
            tablakellek[i, j].BackColor = Color.Black;
        // Közepes
        if (j == 10 && (i >= 2 && i <= 10))
            tablakellek[i, j].BackColor = Color.Black;
        if (j == 2 && (i >= 2 && i <= 10))
            tablakellek[i, j].BackColor = Color.Black;
        if (i == 2 && (j >= 2 && j <= 10))
            tablakellek[i, j].BackColor = Color.Black;
        if (i == 10 && (j >= 2 && j <= 10))
            tablakellek[i, j].BackColor = Color.Black;
        //Kicsi
        if (j == 4 && (i >= 4 && i <= 8))
            tablakellek[i, j].BackColor = Color.Black;
        if (j == 8 && (i >= 4 && i <= 8))
            tablakellek[i, j].BackColor = Color.Black;
        if (i == 4 && (j >= 4 && j <= 8))
            tablakellek[i, j].BackColor = Color.Black;
        if (i == 8 && (j >= 4 && j <= 8))
            tablakellek[i, j].BackColor = Color.Black;
        //Vonalak
        if (j == 6)
            tablakellek[i, j].BackColor = Color.Black;
```

A gombok színezése

```
if (i == 0 && j % 6 == 0)
    tablakellek[i, j].BackColor = Color.Brown;
if (i == 6 && j % 2 == 0)
    tablakellek[i, j].BackColor = Color.Brown;
if (i == 12 && j % 6 == 0)
    tablakellek[i, j].BackColor = Color.Brown;
if (j == 6 && i % 2 == 0)
    tablakellek[i, j].BackColor = Color.Brown;
if ((i == 10 && j == 2) || (i == 2 && j == 2) || (i == 2 && j == 10) || (i == 10 && j == 10))
    tablakellek[i, j].BackColor = Color.Brown;
if ((i == 8 && j == 4) || (i == 8 && j == 8) || (i == 4 && j == 4) || (i == 4 && j == 8))
    tablakellek[i, j].BackColor = Color.Brown;
```

Ez csak egy részlet a tábla színezéséről. Amint lehetett látni a táblán, sajnálatos módon nem tudtam transzparenssé tenni azokat a mezőket, amik fehérek.

Miután beállítottam a színt, ezt a tablakelleket beletettem egy Panelbe aminek a neve simán tabla, és a végén ezt adtam hozzá a Formhoz a Controls segítségével.

Fő osztályok

Három osztályt hoztam létre. Ezek neve Mezo, Babu és Babuk.

Mezo osztály:

A Mezo osztály tartalmaz egy x és egy y koordinátát, illetve egy Babu(neve babu) típusú taggal is rendelkezik. A babu-val tudjuk megnézni későbbiekben, hogy az adott mezőn milyen bábu is szerepel. A konstruktorában beállítom az x és y értékeit, úgy, hogy majd a példányosításnál kapja meg ezeket. Itt hozzáadok egy klikkelés eseményt is. A Mezo klikk eseményében írom meg, hogy hogyan adja hozzá a táblához az adott bábut. Erről későbbiekben írok.

Babu osztály:

A Babu osztályom egy PictureBox. Az osztálynak van egy bool típusú adattagja, ami a bábuk színéért felelős. Ha a szín true, akkor a bábu fehér lesz, ha false, akkor fekete. Ezen kívül a Babu egy Mezo(neve mezo) típusú adattagot is kap. A konstruktorában beállítom, hogy a mezot és a színt példányosításnál lehessen megadni. Ezen felül a mezo segítségével azonnal hozzáadom a kezelőfelülethez → mezo.Controls.Add(this). Itt is létrehozok egy klikk eseményt, mivel azt is le kell kezelnünk, amikor egy bábura kattintunk. A kurzort is itt állítom át „kézre”, így akármikor egy bábu fölé viszi a felhasználó az egeret, kéz lesz az egérből.

Babuk osztály:

Itt állítom be a bábuk színét, illetve itt írom meg azt is, hogy hogyan tud lépni egy bábu. A konstruktoránál úgymond „elkérem” a Babu konstruktorát és a szin alapján bekérem az adott képet. A lépés megvalósításáról később írok.

A Mezo és a Babu adattagjai privátok, ezért ennek a két osztály tagjait get-el és set-el érhetjük el.

Mielőtt leírom a program működését, előtte szükségét érzem, hogy leírjam ezt az információt: A dokumentálásban legtöbbször a fehér bábuk során írom le a program működését. Természetesen a fekete bábukra is ugyanazt a szabályt hoztam létre, mint a fehérre, de legtöbbször a fekete bábuknál nem írom le, hogy hogyan valósítottam meg a rájuk vonatkozó szabályokat, mivel eléggé kicsi az eltérés a két megvalósítás között.

Kezdő Bábuk

A kezdő bábukat egy eljárásban hozom létre, aminek a neve KezdoBabuk. Itt egy for ciklussal 9-szer létrehozok két darab bábut a Babuk osztállyal. Példányosításnál kér egy Mezot és egy bool-t. Ahogy írtam, ha true-t írok be, akkor fehér lesz, ha false-t akkor fekete. A kezdeti mezőre létrehoztam statikusan két Mezo típusú tagot, így a későbbiekben egyszerűbb lesz leellenőrizni, hogy van- e még kezdő bábu. A létrehozott bábukat elmentem két külön Babu listába, hogy később könnyebben le tudjam ellenőrizni őket. A for ciklus után hozzáadom a Formhoz a Controls segítségével.

Alapvető adattagok

tablakellek – Egy 13x13-as Mezo típusú mátrix, ez a táblám.

KezdoFeher – Mezo típus, a fehér játékos kezdő helye.

KezdoFekete - Mezo típus, a fekete játékos kezdő helye.

Add – Mezo típusú lista, azokat a mezőket töltöm bele, amiket később vissza szeretnék színezni.

KijeloltBabu – Babu típusú, akkor rakok bele bábut, ha egy játékos rányom egy bábura.

feherbabuk – Ebben a listában tárolom el a fehér bábukat.

feketebabuk – Ebben a listában tárolom el a fekete bábukat.

dbok – int típusú tömb, fehér bábuk malmainak mentése.

dbok2 – int típusú tömb, fekete bábuk malmainak mentése.

valtas – bool változó, melyik játékos lép.

db – int változó, minden lépés után növekszik.

feherleveheti – bool változó, új malomnál true értéke lesz.

feketeleveheti – bool változó, új malomnál true értéke lesz.

gameover – bool változó, ha true, akkor a játéknak vége.

TextHely – Label típusú, információk megjelenítése (melyik játékos jön, melyik játékos vehet le bábut).

FeherOldal – Label típusú, a „Levett bábuk” feliratért felelős a fehér oldalon.

FeketeOldal - Label típusú, a „Levett bábuk” feliratért felelős a fekete oldalon.

FeherLevettBabuk – PictureBox típusú 3x3-as mátrix, ebből jelenítünk meg annyi képet, amennyi bábut levett a fekete játékos.

FeketeLevettBabuk - PictureBox típusú 3x3-as mátrix, ebből jelenítünk meg annyi képet, amennyi bábut levett a fehér játékos.

Beszorulás vizsgálása

A játék során folyamatosan figyelnie kell a programnak, ha egyik játékos már nem tud sehova se lépni. Megoldásképpen a tablakellekemhez és a formomhoz hozzárendeltem egy mousemove eseményt. Így akármikor a játékos(ok) az egeret a pálya vagy a form részére viszi, akkor ellenőrzést csinál, kiszámolja a bábuk lépési lehetőségeit.

Ez a módszer és a lépések megvalósítása sokban hasonlít egymásra. Nézzük meg a fehér bábuk esetében a program hogyan vizsgálja meg, hogy tudunk-e még lépni.

```
int ox = 0;
int oy = 0;
int dir = 0;
int[] fehersegedt = new int[feherbabuk.Count];
int[] feketesegedt = new int[feketebabuk.Count];
bool fehervereseg = false;
bool feketevereseg = false;
```

Az ox és oy a vizsgált bábu x és y koordinátáját fogja felvenni. A dir segítségével határozom majd meg az irányt. A fehersegedt tömböm, mindig akkora lesz, amennyi bábum van. A fehervereseg ha true értéket vesz fel, akkor tudjuk, hogy a fehérnek nincsen lépési lehetősége.

```
if (!gameover)
{
    if (!FeherLevetel())
    {
        if (valtas)
        {
            for (int i = 0; i < feherbabuk.Count; i++)
            {
                ox = feherbabuk[i].GetMezo.GetX;
                oy = feherbabuk[i].GetMezo.GetY;

                if (db >= 17 | feherbabuk.Count >= 4)
                {
                    dir = 1;
                    while (Tablanbelul(ox, oy + dir) && !BabuEszleles(ox, oy + dir))
                    {
                        if (tablakellek[ox, oy + dir].BackColor == Color.Brown | tablakellek[ox, oy + dir].BackColor == Color.Green)
                        {
                            fehersegedt[i] = 1;
                            break;
                        }
                        if (tablakellek[ox, oy + dir].BackColor == Color.Transparent)
                        {
                            break;
                        }
                        dir++;
                    }
                    dir = 1;
```

Lépési lehetőség vizsgálata a bábutól lefele

Mielőtt el kezdeném vizsgálni az irányokat, előtte biztosítanom kell, hogy a vizsgálások csak akkor menjenek végbe, ha a játék tart, tehát, ha a gameover változóm false. Utána azt is le kell vizsgálnom, ha van malom, mivel lehetséges, hogy amikor malmot alkotok, mindegyik fehér bábu be van szorulva, de miután levett egy bábut lehetséges, hogy utána már tudja mozgatni egyiket. Utána csak azt kell vizsgálnom, hogy melyik játékos van soron. Ha a valtas true, akkor a fehér játékost kell vizsgálnom, mivel neki kell lépnie. Még egy vizsgálat hátravan, mégpedig tudjuk, hogy a játék elején és a játék végén lehetetlen bezárni a másikat, ezért az ellenőrzés csak akkor megy végbe, ha nem lehet ugrálni a bábukkal.

Miután minden szükséges vizsgálat lement, utána egy for ciklussal végig megyek a feherbabuk-on. Az ox és oy-t beállítom az adott bábu x és y koordinátájára. Ezután jönnek az irányok vizsgálása minden bábunál.

A fenti képen az látható, ahogyan a bábutól lefelé vizsgálja meg a program a lépési lehetőségeket. While ciklusom addig megy, amíg a vizsgálatom a táblán belül van, illetve amíg bábut nem talál. Ezen belül, ha talál olyan mezőt, ami barna, illetve itt azt is le kell kezelni, ha zöld (amikor a játékos kiválaszt egy bábut a lehetséges lépési mezőnek a színe zöld lesz), akkor a fehersegedt elemét 1-re állítja. Ugyanígy megvizsgálom a többi irányban is és, ha van lehetséges lépése a bábunak a fehersegedt elemét 1-re állítja.

```
if (db >= 17 | feherbabuk.Count >= 4)
{
    int db = 0;
    for (int i = 0; i < fehersegedt.Length; i++)
    {
        if (fehersegedt[i] == 0)
            db++;
    }
    if (db == feherbabuk.Count)
        fehervereseg = true;
    if (fehervereseg)
    {
        KiJon("Fekete játékos nyert!", Color.Black);
        MessageBox.Show("Fehér nem tud lépni!");
        gameover = true;
    }
}
```

Miután megvizsgálta a program mind a 4 irányt az összes bábunál, utána megnézem, hogy a fehersegedt-ben mennyinél maradt 0 az érték. Minden egyes olyan értéknél, ahol 0 növelek egy db nevű változót, ezután megvizsgálom, hogy ez a db egyenlő-e a bábuim számával. Igaz esetén a fehervereseget true-ra kell állítanom, mivel ez azt jelenit, hogy egyik bábum sem tud ellépni. Ugyanezeket a vizsgálatokat megcsinálom a fekete bábuknál is.

Lépések

A lépéseket virtuális eljárással (Zold) valósítottam meg. A Babu osztályban hoztam létre, viszont a Babuk osztályban írom felül.

```
protected virtual void Zold()
{

}
```

Zold eljárásom a Babu osztályban

```
protected override void Zold()
{
    base.Zold();
    int ox = this.GetMezo.GetX;
    int oy = this.GetMezo.GetY;
    List<Mezo> ZoldLista = new List<Mezo>();
    int dir = 1;
```

Zold eljárásom a Babuk osztályban

.

Segítségül létrehoztam kettő függvényt. A Tablanbelul megvizsgálja, hogy a kijelölt „út”, amit vizsgálni fogok a táblán belül van-e.

A BabuEszleles a vizsgált „útszakaszon” megnézi, hogy van- e bábu. Mindkét függvény vár egy x és egy y koordinátát. A Tablanbelul megvizsgálja, hogy ez az x és y nagyobb-e mint 0 és kevesebb-e mint 13(ez a tábla mérete). A BabuEszleles ezt az x és y tagot a tablakelleknek adja, ezután akkor lesz a függvény true, ha nincsen ott bábu, amit a tablakellek[x,y] koordinátája vizsgál.

```
public static bool Tablanbelul(int x, int y)
{
    return x >= 0 && y >= 0 && x < 13 && y < 13;
}
```

```
public static bool BabuEszleles(int x, int y)
{
    return tablakellek[x, y].GetBabu != null;
}
```

Illetve létrehoztam még két darab listát, az egyik azért kell, hogy be tudjuk színezni zöldre (ZoldLista) a mezőket, a másik, hogy vissza tudjuk állítani (Add). Amikor ráklikkelünk egy bábura, az Add-hoz hozzáadjuk ezt a mezőt, mivel ezt is vissza kell majd állítani. Klikkelésnél a Babuk osztályban a felülírt virtuális függvényben (Zold) le ellenőrzöm, hogy az adott bábu kezdő helyen áll-e vagy sem vagy a listám, amibe eltárolom a bábukat (feherbabuk és feketebabuk) egyenlő-e 3-al, mivel a játékos, akkor is tud ugrálni, amikor 3 bábuja maradt. Ha igen, akkor a felül írt virtuális függvényben for ciklussal végig megyek a táblán és ellenőrzöm, hogy van- e bábu illetve, hogy a mező színe barna-e. Ha igen beletöltöm a ZoldListába ezeket a mezőket.

```
if ((ox == jatek.KezdoFeher.GetX && oy == jatek.KezdoFeher.GetY) || jatek.GetFeher().Count == 3)
{
    for (int i = 0; i < jatek.tablakellek.GetLength(0); i++)
    {
        for (int j = 0; j < jatek.tablakellek.GetLength(1); j++)
        {
            if (jatek.tablakellek[i, j].BackColor == Color.Brown && !jatek.BabuEszleles(i, j))
                ZoldLista.Add(jatek.tablakellek[i, j]);
        }
    }
}
```

Ha a vizsgált bábum nem a kezdőn áll, hanem a táblán, de van még kezdőbábu, akkor addig nem szabad hagyni a játékosnak a táblán álló bábuval lépni, amíg le nem lépte a kezdőbábuit! Ezért a fehér játékos esetében a db változómat megvizsgálom, hogy nagyobb-e 17-nél. Ha igen, akkor engedélyezem a játékos számára a táblán való lépkedést, mivel akkor már biztosan letette az összes bábuját.

Ha a bábu, amire klikkeltem már táblán áll, akkor egy while ciklussal (a felülírt virtuális függvényben) addig megyek, amíg az útvonalam a táblán belül van, vagy amíg bábut nem talált. Az útvonal megtervezéséhez tudni kell a bábu koordinátáit, ezért az ox-et a vizsgált bábumnak az x koordinátájával, az oy-t a pedig az y koordinátájával teszem egyenlővé. Továbbá egy int típusú adattagot (dir) használok, a lehetséges útvonalak felderítésére. Mivel itt egy bábu 4 féle irányba léphet, ezért négy darab while ciklussal kell megnéznem, az adott lépési lehetőségeket. Minden egyes ciklus kezdés előtt 1re kell raknom a dir. Az adott iránytól függően kell az ox-ből vagy az oy-ból kivonnom, vagy hozzáadnom a dir. Fontos még az is, hogy ha talált olyan mezőt a while ciklus, ami barna és nincs rajta bábu, akkor break-el meg kell szüntetni a ciklust. Ha nincs break, akkor a ciklus tovább fog menni, és ha talál még egy ilyen mezőt, azt is zöldre fogja színezni, ami nem jó, mivel a bábukkal csak egyet lehet lépni. Illetve amikor a tábla része transzparens, akkor is breakelem, így kiküszöbölhető az, hogy átlépjen egy másik négyzetre a játékos.

```
dir = 1;
while (jatek.Tablanbelul(ox + dir, oy) && !jatek.BabuEszleles(ox + dir, oy))
{
    if (jatek.tablakellek[ox + dir, oy].BackColor == Color.Brown)
    {
        ZoldLista.Add(jatek.tablakellek[ox + dir, oy]);
        break;
    }
    if (jatek.tablakellek[ox + dir, oy].BackColor == Color.Transparent)
    {
        break;
    }
    dir++;
}
```

Lehetséges útvonal vizsgálása a bábutól jobbra

Miután hozzáadtam azokat a mezőket a ZoldLista-ba, amik megfeleltek, végig megyek a lista elemein és zöldre állítom a színüket. Az Add listához hozzáadom a ZoldLista elemeit. A táblák vissza szÍnezését a Tablaszinreset eljárásban valósítom meg. For ciklussal végig megyek rajta és visszaállítom barnára az elemeit. Ezt az eljárást minden egyes klikkelés után meghívom.

A Mezo osztályban írom meg, azt hogy mi történjen, ha a klikkelés olyan mezőre ment, aminek a színe zöld. Ekkor a Controls segítségével a mezőhöz addolja a KijeloltBabut. A KijeloltBabu egy Babu, ami a klikkelés pillanatában egyenlő lesz azzal a bábuval, amire klikkelt a felhasználó. Miután hozzá addolta a bábut a mezőhöz, az előző mezőn álló bábut nullra állítja, és az új mezőn lévő bábut, beállítja a KijeloltBabura, és ennek a bábunak a mezőjét egyenlővé teszi az új mezővel. Ezután a valtast megfordítja. A valtas változóm egy bool, ami, ha true akkor a fehér léphet, ha false, akkor a fekete.

Levétel

A levételt két külön függvényben írtam meg. Az egyik a fehér bábukat(FeherLevetel) nézi meg, a másik a fekete bábukat(FeketeLevetel). A FeherLevetel függvényben végig megyek a bábukon. A feherbabu listán for ciklussal megvizsgálom, hogy mikor lép egy-egy bábu az adott mezőre. Erre létrehoztam egy bool tömböt, és egyes elemei a mezőt jelzik. Ha valamelyik elem true, az azt jelenti, hogy azon a mezőn áll egy bábu.

```
if (feherbabuk[i].GetMezo.GetX == 0 && feherbabuk[i].GetMezo.GetY == 0)
    feherdbok[0] = true;
if (feherbabuk[i].GetMezo.GetX == 0 && feherbabuk[i].GetMezo.GetY == 6)
    feherdbok[1] = true;
if (feherbabuk[i].GetMezo.GetX == 0 && feherbabuk[i].GetMezo.GetY == 12)
    feherdbok[2] = true;
```

Ezután a tömb elemeinél megnézem, hogy az adott 3 mezőn, ami malmot képez mind a három érték true-e. Ha igen, akkor egy másik int típusú tömb (dbok) értékét növelem. A végén egy for ciklussal megnézem a dbokat és, ha talál benne 1-est, akkor true-ra állítja a feherlevehetit. Amikor megszakad a malom, akkor a hozzárendelt dbok értékét 0-ra állítja. Így akármikor a játékos újra malmot képez azon a részen, a hozzárendelt dbok értéke 1 lesz. Ha a malom sokáig van fent tartva, akkor szimplán növeli, addig, amíg meg nem szünteti a malmot.

```
if (feherdbok[0] && feherdbok[1] && feherdbok[2]
    dbok[0]++;
else
    dbok[0] = 0;
```

Megvizsgáljuk a 3 mezőt

```
for (int i = 0; i < dbok.Length; i++)
{
    if (dbok[i] == 1)
        leszed = true;

}
return leszed;
```

Megvizsgáljuk van-e 1-es értékű elem a tömbben

A FeherLevetelt minden egyes lépés után ellenőriz a program. Ha feherleveheti, vagy a feketeleveheti true, akkor a Babuban lévő klikkelés megy végbe, az ottani klikkelés felelős a bábuk levételéért. Amikor bekövetkezik a levétel, megvizsgálja a Babu klikkelésében, hogy amire kattintunk az valóban bábu-e. Majd megnézi, hogy a bábu színe megegyezik-e a valtassal. Ha igen akkor elméletileg le lehet venni a bábut, mivel amikor true lesz valamelyik leveheti, a valtas változó még előtte átvált az ellenfél színére, tehát a levételt, csak akkor szabad engedélyezni, ha a valtas és a bábu színe megegyezik. Azért írtam, hogy elméletileg mivel, ezután ellenőrzöm, hogy melyik bábut szeretné le venni a játékos. Most azt az esetet fogom leírni, amikor le tudjuk venni az adott bábut, a következő fejezetek pedig arról fognak szólni, hogy hogyan kezeltem le azt, ha a játékos a malom szabályainak nem megfelelő bábut

szeretne levenni. Megtörténik a malom, és tételezzük fel, hogy a játékos olyan bábut szeretne levenni, ami nincsen malomban és az ellenfél kezdőhelyén sem, akkor a Controls.Remove segítségével, eltüntetjük a bábut, a mezőjét és bábuját nullra állítjuk, hogy később oda lehessen lépni. Ezután eltávolítom a listámból az adott bábut, kiírom a játékosnak melyikük léphet és le is ellenőrzöm, hogy a listák hossza nem egyenlő-e kettővel. Ha igen, úgy a játéknak vége. Ezután a levehetiket false-ra állítom, hogy a játék tovább vigye magát. Majd végül meghívom a BabuLevetelPic-et, ami a levett bábuk képeiért felelős, de erről később írok majd egy másik fejezetben.

```
this.mezo.Controls.Remove(this);
this.mezo.SetBabu = null;
this.mezo = null;
if (this.szin == true)
    jatek.GetFeher().Remove(this);
else
    jatek.GetFekete().Remove(this);
jatek.KiJon(jatek.valtas ? "Fehér játékos lép!" : "Fekete játékos lép!", jatek.valtas ? Color.White : Color.Black);
if (jatek.GetFeher().Count == 2 || jatek.GetFekete().Count == 2)
    jatek.gameover = true;
if (jatek.GetFeher().Count == 2)
    jatek.KiJon("Fekete játékos nyert!", Color.Black);
if (jatek.GetFekete().Count == 2)
    jatek.KiJon("Fehér játékos nyert!", Color.White);
jatek.feherleveheti = false;
jatek.feketeleveheti = false;
jatek.BabuLevetelPic();
```

Bábu eltávolítása

Egy játék során felmerülhetnek olyan játékállások, amikor egy játékos nem veheti le a kiszemelt bábuját, mivel az a bábu malmot képez. Viszont olyan esetek is vannak, amikor az ellenfél bábuinál nincsen szabad bábu, tehát mindegyik malomban áll. Ilyenkor levehetünk a malomból lévő bábukból.

GameGoOn és IllegalPick

Ha egy játékosnak sikerült malmot csinálni, akkor utána lefut a GameGoOn függvény ami

leellenőrzi, hogy van-e levehető bábu a pályán. Ha van, akkor megy tovább és leellenőrzi az IllegalPick-el, hogy amilyen bábura kattint a játékosunk azt le lehet-e venni , tehát malmot alkot-e a kiválasztott bábu.

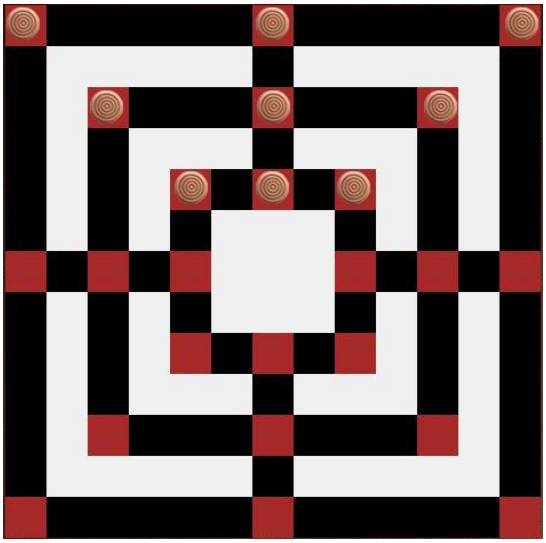
GameGoOn: Ez a függvényem felelős azért, hogy leellenőrizze, hogy tudunk-e malomból levenni. Ez csak akkor lehetséges, ha csak malomban lévő bábukból lehet választani az ellenfél bábui közül. Alapját tekintve úgy gondoltam, hogy ezt a problémát egyszerűen el lehet intézni, ha a bábuimnak adok egy bool adattagot és azoknak a bábuknak, amelyek malmot alkotnak ezt az adattagot true-ra rakom, ha létrejött malom. Ha van olyan bábu a pályán, ami nincs malomban annak ez az adattagja false. Így csak annyit kellene leellenőrizni, hogy van-e a pályán olyan bábu, aminek ez az adattagja false. Ha van, akkor ez a függvény hamis értékkel tér vissza. Viszont ez a módszer nem működött, ezért sajnos bonyolultabban kellett megoldanom ezt a problémát.

Sokáig gondolkodtam hogyan lehetne ezt a szabályt lekezelni, amikor rájöttem, hogy ha van x adott bábum a pályán és van y db malmom, akkor biztosan vannak olyan játékesetek, amikor csak malom lehet a pályán. Például ha már csak 3 bábu van a pályán és a malmok száma 1,

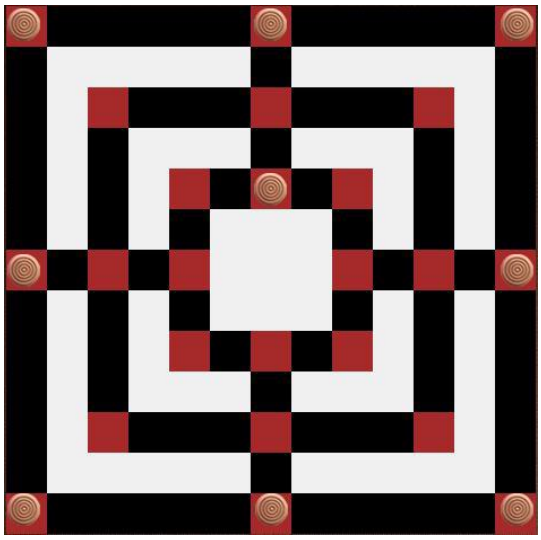
akkor biztosan nem lesz olyan bábu, amit a játékos levehet, tehát engedélyezni kell a malomból történő levételt. Ám ez visszafele is igaz, tehát, ha van 4 bábum és a malmok száma 1 (több nem is lehet), akkor biztosan lesz olyan bábu, amit le lehet venni és nem malomból. Erre a gondolatmenetre alapozva jött létre az a megoldásom, hogy for ciklussal végig megyek a bábukat tartalmazó listákon, és ha ezeknek az x koordinátájuk nem egyenlő a kezdő mezővel, akkor a pályán vannak. A „babupalyan" változómat növelem, ha van ilyen bábu. Így megkapom hány bábu tartózkodik a pályán. Utána egy for ciklussal végig megyek a „dbok"-on. Ha bármelyiknek az értéke nagyobb mint 0, akkor tudjuk, hogy valahol a pályán malom van. Ekkor növelem a „malmokszama" változómat.

Ezután megvizsgálom, hogy hány bábu és hány malom van a pályán. Vannak olyan esetek, amikor van x db bábu a pályán és y db malom és van egy olyan bábu, ami nem áll malomban, de van olyan eset, amikor az összes pályán lévő bábu malomban van. Példán keresztül most bemutatok egy ilyet:

Van 9db bábu a pályán. 9 bábuval a maximális malomszám 4 lehet. Viszont ez a 4 malom kijöhet úgy is, hogy marad egy nem malomban lévő bábu, de kijöhet úgy is, hogy mindegyik bábut felhasználjuk.



Amikor felhasználjuk az összes bábut



Amikor marad egy nem malomban lévő bábu

Tehát először meg kell nézni, hogy melyek azok az esetek, amikor egyáltalán ellenőrzést kell végeznünk.

```
if (babupalyan == 9 && malmokszama == 4)...
else if (babupalyan == 9 && malmokszama == 3)...
else if (babupalyan == 8 && malmokszama == 4)
    levehet = true;
else if(babupalyan == 8 && malmokszama == 3)...
else if (babupalyan == 7 && malmokszama == 3)
    levehet = true;
else if (babupalyan == 6 && malmokszama == 2)...
else if (babupalyan == 5 && malmokszama == 2)
    levehet = true;
else if (babupalyan == 3 && malmokszama == 1)
    levehet = true;
```

Az összes olyan eset, amikor ellenőrzést kell végezni

Ilyen esetek azok, amikor 9 bábu van a pályán és 4 malom van, vagy amikor 9 bábu van a pályán és 3 malom stb. Szépen végig mentem az összes ilyen leellenőrzésen. Egyszerűbb volt azokat az eseteket vizsgálni, amikor maradni fog bábu. Maradjunk ennél a példánál, amikor 9 bábu van és 4 malom. Itt 3 olyan lehetséges állás létezik, amikor marad is egy bábu. Ez a 3 lehetséges esetek azok, amikor a bábuk úgy helyezkednek el, hogy vagy a felső mezőknél alkotnak 4 malmot, vagy a középsőnél, vagy a kicsinél. Az összes többi esetnél nem fog maradni levehető bábu! Ha a függvényt nézzük, akkor „babupalyan == 9 && malmokszama == 4 „ vizsgáláson belül, megnézem, hogy a felső mezőkhöz tartozó dbok adattag nagyobb-e mint 0. Ha igen, akkor tudjuk, hogy 1 bábu maradt még, amelyik nincs malomban, tehát a GameGoOn false értékkel tér majd vissza.

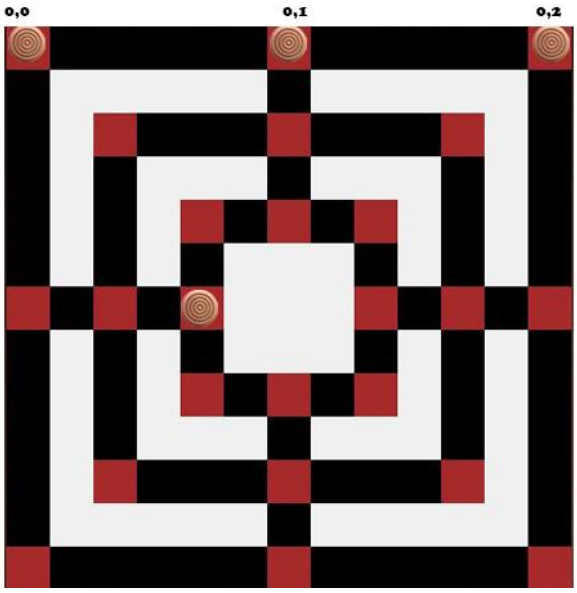
```
if (babupalyan == 9 && malmokszama == 4)
{
    if (dbok[0] > 0 && dbok[1] > 0 && dbok[2] > 0 && dbok[3] > 0)
        levehet = false;
    else if (dbok[4] > 0 && dbok[5] > 0 && dbok[6] > 0 && dbok[7] > 0)
        levehet = false;
    else if (dbok[8] > 0 && dbok[9] > 0 && dbok[10] > 0 && dbok[11] > 0)
        levehet = false;
    else
        levehet = true;
}
```

Ezeknek a lehetséges lépéseknek a leellenőrzéséhez papírt és bábukat használtam, úgymond eljátszottam az összes olyan lehetséges eseteket, amikor maradni fog egy bábu és azokat is, amikor nem. Ez a módszer nem a legbiztonságosabb, hiszen napról napra jöttem rá újabb lehetséges esetekre, viszont amit a fejezet elején írtam lehetséges megoldást az nem vált be, ezért kénytelen voltam egy kicsit jobban eljátszadozni a bábukkal. 95%-ban biztos vagyok benne, hogy mostanra már az összes olyan esetet lekezel a program, ami lehetséges egy játékon belül. Ha mégsem, akkor akármikor bele írhatom az újabb eseteket.

Mi történik akkor, ha a GameGoOn true? Akkor megy tovább úgy a játék, hogy az IllegalPick függvényt nem hívom meg, tehát bármelyik bábut le lehet venni. Ha viszont false lesz, akkor belemegy az IllegalPickbe.

IllegalPick: Miután megbizonyosodtunk arról, hogy van levehető bábu, akkor meg kell vizsgálnunk azt, hogy ha a játékos olyan bábut szeretne levenni, ami malomban van, azt ne engedjük neki, hiszen van levehető bábu.

Az IllegalPick kér egy bábut majd a meghívásakor, ami az a bábu lesz, amit a játékos éppen le szeretne venni. A függvényben megnézem ennek a bábunak a koordinátáját. Elég lesz csak az egyiket nézni, mivel így is be tudjuk azonosítani, hogy ha áll is malomban akkor melyikben. Mivel példával jobban szeretem bemutatni, ezért nézzünk itt is egy példát:



A képen látható a fehér bábuk elhelyezkedése. A fekete játékos le szeretné venni a 0,0 mezőn álló bábut. Ekkor az IllegalPick-ben switch, case-el megnézem a bábu x koordinátáját. Ha az 0, akkor megnézem a hozzá tartozó malmot a dbok-ban és, ha az nagyobb mint 0, akkor tudjuk, hogy ez a bábu malomban van. Ha a bábunak az x koordinátáját nézem, akkor mindig a vízszintes mezőket nézem, ha az y koordinátát, akkor pedig a függőleges mezőket nézi.

Mentés és betöltés

Alapvetően adatbázisban tároltam volna el azokat az adatokat, amiket le kell menteni a játék során. Többszöri sikertelen próbálkozás után úgy döntöttem, hogy ez a módszer a további fejlesztések során kerül bele a programba. Addig is a mentést a SaveFileDialog segítségével és fájlba írással oldottam meg.

A SaveFileDialog csak txt-t enged lementeni,

A fájlba írás során lementjük, hogy melyik játékos lép (valtas), utána a két false mutatja, hogy melyik játékos vehet le bábut (feherleveheti,feketeleveheti). Alapvetően a játékos nem tud menteni miközben a levétel folyamatban van (későbbi fejlesztés), így ezt a két adattagot feleslegesen írom a fájlba. Későbbiekben ezt ki szeretném majd javítani, ezért beleírom. Utána következnek a bábuk koordinátái (x,y), majd a bábuk színe (szin). Az utolsó sorban végül elmentem a dbok-t és a dbok2-t. Erre azért van szükség, hogy a játék megkapja, azokat a mezőket, ahol éppen malomban állnak a bábuk.

```
True-False-False
86 287 True
86 287 True
86 287 True
86 287 True
86 287 True
86 287 True
86 287 True
86 287 True
86 287 True
783 287 False
783 287 False
783 287 False
783 287 False
783 287 False
783 287 False
783 287 False
783 287 False
783 287 False
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Betöltésnél meg kell vizsgálnunk, hogy a felhasználó jó adatot tölt-e be, ha nem akkor kiír egy hibaüzenetet a program. A fájl típusa, ha rendben van és nem hiányzik semmi a fájlból, akkor StreamReader segítségével beolvasom a fájl tartalmát. Az egesz adattagomban tárolom, a fájl összes elemét, majd az utolsosor tagommal eltárolom a fájl utolsó sorát. Az első sor karaktereit az elsosor tömbömben tárolom el, majd a játék változóit (valtas,feherleveheti,feketeleveheti) egyenlővé teszem a tömb adott elemeivel.

```csharp
Babureset();
BabuPicClear();
StreamReader file = new StreamReader(load.FileName);
string[] egesz = File.ReadAllLines(load.FileName);
string utolsosor = egesz[egesz.Length - 1];
string[] seged = utolsosor.Split();
string sor = file.ReadLine();
string[] elsosor = sor.Split('-');
valtas = bool.Parse(elsosor[0]);
feherleveheti = bool.Parse(elsosor[1]);
feketeleveheti = bool.Parse(elsosor[2]);
```

Ezután for ciklussal az első sort kihagyva végigmegyek az utolsó előtti sorig. Ahogy megyek végig a fájlon a sor változómban eltárolom a fájl sorait, majd ezt a segedtomb tömbömben space alapján splitelve eltárolom. Ebben a tömben lesznek a bábuk koordinátái és színei(segedtomb[0] = a bábu x koordinátája,segedtomb[1] = a bábu y koordinátája,segedtomb[2] = a bábu színe). Ezután swtich case segítségével megvizsgálom a segedtomb 2. indexét, ami a fájlban a bábu színei. Ha True-t talál, akkor tudjuk, hogy az a bábu fehér, viszont utána meg kell néznünk azt is, hogy a táblára kell-e lehelyezni a bábut vagy a kezdőre. Itt szimplán if-el megnézem, hogy a segedtomb 0.indexén lévő szám megegyezik-e a kezdőmezőnek az x koordinátájával. Ha igen, akkor új bábu példányosításánal a konstruktornak megadom a kezdőt és a segedtomb 2.indexét, majd ezután hozzáadom a feherbabu listához. Ha hamis, akkor a példányosításnál, ahol vár egy Mezo típust, a tablakelleknek a segedtomb 0. és 1. indexén lévő koordinátát adom meg, majd utána, ahol a színét kéri megadom a segedtomb 2. indexén lévő értéket. Ezzel a módszerrel ugyanúgy leellenőrzöm a fekete bábuknál is.

```
for (int i = 1; i < egesz.Length-1; i++)
{
    sor = file.ReadLine();
    string[] segedtomb = sor.Split();

    switch(segedtomb[2])
    {
        case "True":
            if(int.Parse(segedtomb[0]) == KezdoFeher.GetX)
            {
                Babuk feher = new Babuk(KezdoFeher, bool.Parse(segedtomb[2]));
                feherbabuk.Add(feher);
            }
            else
            {
                Babuk feher = new Babuk(tablakellek[int.Parse(segedtomb[0]), int.Parse(segedtomb[1])], bool.Parse(segedtomb[2]));
                feherbabuk.Add(feher);
            }
            break;
        case "False":
            if(int.Parse(segedtomb[0]) == KezdoFekete.GetX)
            {
                Babuk fekete = new Babuk(KezdoFekete, bool.Parse(segedtomb[2]));
                feketebabuk.Add(fekete);
            }
            else
            {
                Babuk fekete = new Babuk(tablakellek[int.Parse(segedtomb[0]), int.Parse(segedtomb[1])], bool.Parse(segedtomb[2]));
                feketebabuk.Add(fekete);
            }
            break;
    }
}
```

Miután a bábuk betöltődtek, a dbok-ot és a dbok2-t kell beállítani. For ciklussal bejárom a dbok-ot és egyenlővé teszem a seged i. elemével. Mivel egy tömbben tárolom a dbokat és a dbok2-t is, ezért valahogy szét kell választani őket, hogy a dbok2 is megkapja a megfelelő értékeket. For ciklussal 16-tól kezdve elmegyek a seged-1 –ig (az utolsó érték a db-om lesz, ami számolja, hogy hány bábulerakás történt már), majd ezeknek az értékeit eltárolom egy listába. Ezután for ciklussal végig megyek a dbok2-n és egyenlővé teszem a lista elemeivel.

```
for (int i = 0; i < dbok.Length; i++)
{
    dbok[i] = int.Parse(seged[i]);
}
List<string> fek = new List<string>();
for (int i = 16; i < seged.Length-1; i++)
{
    fek.Add(seged[i]);
}
for (int i = 0; i < dbok2.Length; i++)
{
    dbok2[i] = int.Parse(fek[i]);
}
db = int.Parse(seged[32]);
file.Close();
```

Miután a malmok is betöltődtek, már csak a form megjelenítését kell beállítani. Különféle ellenőrzésekkel megnézem, hogy mit kell kiírni és, hogy mit kell megjeleníteni a felhasználó számára.

```
if (feketeleveheti)
    LevetelKiir("Fekete játékos levehet egy bábut!", Color.Black);
else if (feherleveheti)
    LevetelKiir("Fehér játékos levehet egy bábut!", Color.White);
else
    KiJon(valtas ? "Fehér játékos lép!" : "Fekete játékos lép!", valtas ? Color.White : Color.Black);

if (db <= 16)
{
    KezdoFeher.Visible = true;
    KezdoFekete.Visible = true;
}
if(db >= 17)
{
    KezdoFeher.Visible = false;
}
if(db >= 18)
{
    KezdoFekete.Visible = false;
}
switch (feherbabuk.Count)...// Mennyi képet kell megjeleníteni a levett bábuknál a fehér oldalon
switch (feketebabuk.Count)...// Mennyi képet kell megjeleníteni a levett bábuknál a fekete oldalon
```

Mivel nem csak a játék felületén lehet betölteni, hanem a menüből is, ezért a két betöltés annyiban tér el egymástól, hogy amikor a menüből töltünk be játékot, akkor valahogy el kell érnem a játékban lévő adattagokat is. Ezért létrehoztam olyan függvényeket és eljárásokat, hogy ezeket a tagokat el tudjuk érni és módosítani is tudjuk. A menüben lévő betöltés csak annyibban tér el, hogy közvetlenül nem mindegyik adattagot tudjuk elérni, ezért Set és Get függvénnyel és eljárással tudjuk meghívni, illetve módosítani az adott adattagokat.

Eljárások és függvények

Ebben a fejezetben lesz pár olyan függvény, illetve eljárás, amiről már írtam, de úgy gondoltam, ha összeszedem az összes függvényt és eljárást egy külön fejezetbe és röviden leírom a funkciójukat, akkor átláthatóbb lesz a program működése.

Eljárások:

Tablaletrehozas – Ebben az eljárásban készítem el a táblát.

Tablaszinreset – A mezők visszaállítása az eredeti színére.

KezdoBabuk – Kezdő bábuk lehelyezése a kezdő helyekre, listák feltöltése a bábukkal, malmok (dbok,dbok2) nullázása, db nullázása.

Babureset – Eltüntet minden bábut a tábláról, törli a bábukat tartalmazó listák elemeit, a kezdő helyekről is eltünteti a bábukat.

BabuPicClear - Láthatatlanná teszi a levett bábuk elemeit.

FeherOldalL – Beállítja a FeherOldal tulajdonságait, feltölti a FeketeLevettBabuk-at képekkel és el is helyezi őket a formon. Alapból láthatatlanná teszi a mátrix elemeit.

FeketeOldalL – Ugyanazt csinálja amit a FeherOldalL csak a másik oldalon.

BabuLevetelPic – Ez az eljárás felelős a levett bábuk képeinek helyes megjelenítéséért. Ellenőrzi a bábuk listáját. Akárhányszor valamelyik lista eleme csökken, úgy teszi láthatóvá az adott mátrix elemét.

KiJon – Kiírja, hogy éppen melyik játékos léphet. A valtas alapján írja ki, hogy a fehér vagy a fekete játékos lép-e. A statikus TextHely szövegét, állapotát és helyét változtatja meg.

LevetelKiir – Amikor valamelyik leveheti true, akkor kiirja, hogy melyik játékos vehet le egy bábut. Ez is a statikus TextHely tulajdonságait változtatja meg.

Mentes – A játék mentését hajtja végre.

Betoltes – A játék betöltését hajtja végre.

FormClick – Amikor a játékos kattint a Formra visszaállítja a mezők eredeti színét a Tablaszinresettel

Babureset – For ciklussal végig megyek a táblán és a mezőkről eltávolítok mindent, utána a mezők bábuját nullra állítom. A bábukat tartalmazó listákat kiürítem, a kezdő mezőket is kinullázom és törlöm a rajta lévő bábukat.

Függvények:

Tablanbelul – Megvizsgálja, hogy az adott útvonal a táblán belül van-e.

BabuEszleles – Megvizsgálja, hogy az adott útvonalon van-e bábu.

IllegalPick – Le ellenőrzi, hogy az adott bábu levehető-e.

GameGoOn – Engedélyezi a játékost malomban lévő bábu levételére.

FeherLevetel – Ellenőrzi, hogy malmot alkotnak-e a fehér bábuk.

FeketeLevetel - Ellenőrzi, hogy malmot alkotnak-e a fekete bábuk.

További függvények és eljárások:

Az itt felsorolt függvények és eljárások, csak azt a célt szolgálják, hogy egy másik formból is el lehessen érni a szükséges adatokat:
GetFeher,GetFekete,GetDbok,GetDbok2,SetDbok,SetDbok2.