**Beadandó feladat dokumentáció**

**Készítette:**

Németh Dominik Kristóf

E-mail: [nemethdominik02@gmail.com](mailto:nemethdominik02@gmail.com)

**Feladat:**

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk. Adott egy n × n mezőből álló tábla, amelyre a játékosok 2 × 1-es méretű téglalapokat helyezhetnek el (vízszintesen, vagy függőlegesen). A játékosok felváltva léphetnek. A játék célja, hogy a téglalapokkal elhatároljuk a tábla egy részét (teljesen körbevéve téglalapokkal), amelyben így minden mező a játékosé lesz (beleértve az ellenfél által korábban elfoglalt mezőket is). A program külön jelölje meg a lehelyezett téglalapokat, illetve az elfoglalt területeket, és játék közben folyamatosan jelenítse meg az elfoglalt terület méretét játékosonként. A játék akkor ér véget, ha már további lépés nem végezhető a pályán. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6 × 6, 8 × 8, 10 × 10), valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. Tipp: a bekerítés ellenőrzéséhez használható például az elárasztásos kitöltés (flood fill) algoritmusa.

**Elemzés:**

* A feladatot .NET MAUI alkalmazásként, elsődlegesen Windows és Android platformon valósítjuk meg. Az alkalmazás négy lapból fog állni.
* A játék négy képernyőn fog megjelenni.
  + Az első képernyő (Játék) tartalmazza a játéktáblát, a játék állását (piros, kék pontok) a lap alján, az új játék, valamint a beállítások gombjait a lap tetején, alatta közvetlenül a 2x1-es blokkok lehelyezését véglegesítő gombot.
  + A második képernyőn van lehetőség betöltésre, illetve mentésre, valamint a játéktábla méretének állítására (három kapcsolóval).
  + A további két képernyő a betöltésnél, illetve mentésnél megjelenő lista, ahol a játékok elnevezése mellett a mentés dátuma is látható. Mentés esetén ezen felül lehetőség van új név megadására is.
* A játékot három táblamérettel játszhatjuk: kicsi (6x6), közepes (8x8), nagy(10x10). A program indításakor közepes táblaméretet állít be, és automatikusan új játékot indít.
* A játéktáblát érintéssel kezelhetjük. Az érintés hatására 2x1-es blokkokat helyezhetünk le, amelyet a táblán szereplő gombbal véglegesíthetünk.
* A játék automatikusan jelzi előugró üzenettel, ha vége a játéknak.

**Felhasználói estek diagramja**

A képen diagram, szöveg, kör, képernyőkép látható

Automatikusan generált leírás

**Tervezés:**

* Programszerkezet:
  + A szoftvert két projektből építjük fel: a modellt és a perzisztenciát tartalmazó osztálykönyvtárból (.NET Standard Class Library), valamint a .NET MAUI többplatformos projektből, amelyet Windows és Android operációs rendszerre is le tudunk fordítani.
  + A programot MVVM architektúrában valósítjuk meg, ennek megfelelően **View, Model, ViewModel** és **Persistence** névtereket valósítjuk meg az alkalmazáson belül.
  + A megvalósításból külön építjük fel a játék, illetve a betöltés és mentés funckiót, valamennyi rétegben. Utóbbi funkcionalitást újrahasznosítjuk egy korábbi projektből, így nmem igényel újabb megvalósítást.
  + A program vezérlését az alkalmazás osztály **(App)** végzi, amely példányosítja a modellt, a nézetmodellt és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.

A képen szöveg, diagram, Betűtípus, képernyőkép látható

Automatikusan generált leírás

* Perzisztencia:
  + Az adatkezelés feladata a táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  + A **BekeritesTable** egy érvényes játéktáblát biztosít (azaz mindig ellenőrzi a beállított értékeket), ahol minden mezőre ismert az értéke (**Player**). A tábla alapértelmezés szerint 8x8-as, de ez GameMode választásával állítható. A tábla lehetőséget ad az állapotok lekérdezésére (**IsFilled**, **GetValue**, **Size**, **CurrentPlayer**), valamint szabályos léptetésre (**StepValue**), illetve direkt beállítás (**SetValue**) elvégzésére.
  + A hosszú távú adattárolás lehetőségeit az **IBekeritesDataAccess** interfész adja meg, amely lehetőséget ad a tábla betöltésére (Load), valamint mentésére (Save). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
  + Az interfészt szöveges fájl alapú adatkezelésre a **BekeritesFileDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **BekeritesDataException** kivétel jelzi.
  + A program az adatokat szöveges fájlként tudja eltárolni, amelyeket egy megadott könyvtárban (**\_directory**) helyez el. Ez majd az alkalmazás platformfüggő saját adatkönyvtára lesz.
  + A fájl első sora megadja a tábla méretét, valamint a jelenlegi játékost (**CurrentPlayer**) (ami alapértelmezés szerint 8x8 és piros játékos. A fájl többi része izomorf leképezése a játéktáblának, amely soronkra és oszlopokra bontva számokat tartalmaz szóközökkel választva. A számok 0-3 közöttiek lehetnek, ahol 0 reprezentálja a NoPlayert, 1-2 pedig a piros és a kék játékos.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

* Modell:
  + A modell lényegi részét a **BekeritesGameModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit. A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (StepGame, DealWithSurrounded, SecondBlock).
  + A mezők állapotváltozásáról a **RotatedSecondBlock** esemény tájékoztat. Az esemény argumentuma (**FieldChangedEventArgs**) tárolja megváltozott mező pozícióját.
  + A játékállapot megváltozásáról (elfoglalt terület) a **GameAdvanced** esemény, míg a játék végéről döntetlen esetében a **GameOver** ha van nyertes akkor a **GameWon** esemény tájékoztat. Az események argumentuma (**GameWonEventArgs**) tárolja a győztest.
  + A modell példányosításakor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGame**) és mentésre (**SaveGame**)
  + A játékmódot a **GameMode-on** át kezeljük, a játékosokat pedig a **Player** felsorolási típuson át kezeljük.

A képen szöveg, képernyőkép, dokumentum, Betűtípus látható

Automatikusan generált leírás

* Nézetmodell:
  + A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.
  + A nézetmodell feladatait a **BekeritesViewModel** osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, valamint a kilépéshez. A parancsokhoz eseményeket kötünk, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását (**\_model**), de csupán információkat kér le tőle, illetve a játékmódokat szabályozza. Direkt nem avatkozik a játék futtatásába.
  + A játékmező számára egy külön mezőt biztosítunk (**BekeritesField**), amely eltárolja a pozíciót, szöveget, engedélyezettséget, valamint a lépés parancsát (**StepCommand**). A mezőket egy felügylet gyűjteménybe helyezzük a nézetmodellbe (**Fields**).
  + A lehetséges táblaméreteket egy-egy **GameModeViewModel** objektum reprezentálja, amelyeket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (**MapSizeLevels**).
  + A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

    Automatikusan generált leírásA tárolt játékállapotok egy-egy **StoredGameViewModel** példánnyal írhatóak le. Ezek kollekcióját nem ágyazzuk be a fő nézetmodellbe **(BekeritesViewModel)**, hanem a betöltéskor és mentéskor dinamikusan állítjuk elő és adjuk át a nézet számára.

• Nézet:

• A nézetet navigációs lapok segítségével építjük fel.

• A **MainPage** osztály tartalmazza a játéktáblát, amelyet egy Grid segítségével valósítunk meg, amelyben Button elemeket helyezünk el.

• A **SettingsPage** osztály tartalmazza a betöltés, mentés gombjait, illetve egy Picker példányt a nehézség állítására. • A LoadPage és a SavePage szolgál egy létező játékállapot betöltésére, illetve egy új mentésére.

• Vezérlés (6. ábra):

• Az **App** osztály feladata az alkalmazás vezérlése, a rétegek példányosítása és az események feldolgozása.

• A **CreateWindow** metódus felüldefiniálásával kezeljük az alkalmazás életciklusát a megfelelő eseményekre történő feliratkozással. Így az alkalmazás felfüggesztéskor (**Stopped**) elmentjük az aktuális játékállást (**SuspendedGame**), míg folytatáskor vagy újraindításkor (**Activated**) pedig folytatjuk, amennyiben történt mentés.

• Az alkalmazás lapjait egy **AppShell** keretben helyezzük el. Ez az osztály felelős a lapok közötti navigációk megvalósításáért.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

* **Tesztelés:**
  + A modell funcionalítása egységtesztek segítségével lett ellenőrizve a **BekeritesGameModelTest** osztályban.
  + Az alábbi tesztesetek kerültek megvalósításra:
    - **BekeritesGameModelNewGameSixTest,**
    - **BekeritesGameModelNewGameEightTest**,
    - **BekeritesGameModelNewGameTenTest**: Új játék indítása, adott méretű táblával.
* **BekeritesGameModelStepTest:** Játékbeli lépés hatásainak ellenőrzése, lépés végrehajtás, mező foglalás.
* **BekeritesGameModelLoadTest:** A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.