**3. Beadandó feladat dokumentáció**

Készítette: Börzsei Marcell László

Neptun-kód: BQHFLX

E-mail: [borzseimarcell032@gmail.com](mailto:borzseimarcell032@gmail.com)

**Feladat:**

**Kitolás**

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy 𝑛 × 𝑛 mezőből álló tábla, amelyen kezdetben a játékosoknak 𝑛 fehér, illetve 𝑛 fekete kavics áll rendelkezésre, amelyek elhelyezkedése véletlenszerű. A játékosok kiválaszthat egy saját kavicsot, amelyet függőlegesen, vagy vízszintesen eltolhat. Eltoláskor azonban nem csak az adott kavics, hanem a vele az eltolás irányában szomszédos kavicsok is eltolódnak, a szélső mezőn lévők pedig lekerülnek a játéktábláról. A játék célja, hogy adott körszámon belül (5𝑛) az ellenfél minél több kavicsát letoljuk a pályáról (azaz nekünk maradjon több kavicsunk a végére). Ha mindkét játékosnak ugyanannyi marad, akkor a játék döntetlen. A program biztosítson lehetőséget új játék kezdésére a táblaméret (3 × 3, 4 × 4, 6 × 6) és így a lépésszám (15, 20, 30) megadásával, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött (ha nem döntetlen), majd automatikusan kezdjen új játékot. Ezen felül legyen lehetőség a játék elmentésére, valamint betöltésére.

**Elemzés:**

* A játékot három játékmóddal játszhatjuk: 3x3-as (15 lépés, 9 generált mező), 4x4-es (20 lépés, 16 előre generált mező), 6x6-os (30 lépés, 36 előre generált mező).
* A feladatot Maui alkalmazás segítségével valósítjuk meg
* A játék négy képernyőn fog megjelenni. Az első képernyő (Játék) tartalmazza a játéktáblát, a játék állását (lépések száma, fennmaradó idő) a lap alján, az új játék, valamint a beállítások gombjait a lap tetején. A második képernyőn van lehetőség betöltésre, illetve mentésre, valamint a játéknehézség 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 játéktáblát egy 3x3-as vagy 4x4-es vagy 6x6-os színes négyzetekből álló rács reprezentálja.
* Mindig a fekete játékos kezdi a játékot. A játékban egy label jelzi az éppen kijelölt kavicsot, amellyel lehet lépni, a kijelölést pedig a gomb nyomásával tudjuk változtatni. Majd, ha kiválasztottuk a megfelelő kavicsot, amellyel szeretnénk lépni, akkor a gombok lenyomásával jelezhetjük, hogy milyen irányba szeretnénk lépni.

A képen diagram, kör, vázlat, rajz 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., ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést. A program csomagszerkezete a 2. ábrán látható.
* A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül.
* A megvalósításból külön építjük fel a játék, illetve a betöltés és mentés funkciót, valamennyi rétegben. Utóbbi funkcionalitást újrahasznosítjuk egy korábbi projektből, így nem igényel újabb megvalósítást.
* A program vezérlését az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést
* A képen szöveg, diagram, Tervrajz, Műszaki rajz látható

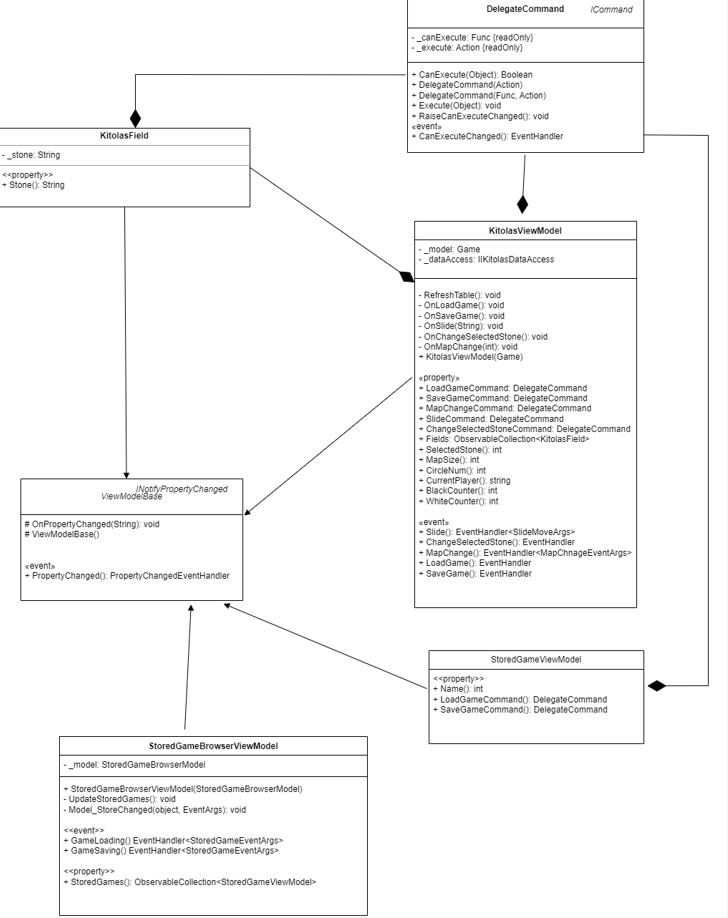
  Automatikusan generált leírás**Perzisztencia:**
* Az adatkezelés feladata a SlideTable táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
* A SlideTable osztály egy érvényes táblát biztosít (azaz mindig ellenőrzi a beállított értékek), ahol minden mezőre ismert az értéke. Magukat az értékeket a tábla létrehozásakor véletlen generáljuk. A táblából alapértelmezés szerint 3x3-as, 4x4-es, illetve 6x6-os hozható létre, és ez a konstruktorban paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére (a különböző fieldeknek a getterei), valamint szabályos léptetésre (slide).
* A hosszú távú adattárolás lehetőségeit az IKitolasDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
* Az interfészt szöveges fájl alapú adatkezelésre a KitolasFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a KitolasDataException 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 program az adatokat szöveges fájlként tudja eltárolni, melyek az stl kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
* A fájl első sora megadja a tábla méretét, a hátralévő körök számát, a fekete majd a fehér kavicsok számát, illetve a soron következő játékost. A fájl többi része izomorf leképezése a játéktáblának, azaz összesen 3, 4 vagy 6 sor következik (játéktábla méretétől függően), és minden sor 3, 4 vagy 6számot tartalmaz szóközökkel választva. A számok 0, 1 vagy 2 lehetnek, ahol 0 reprezentálja az üres mezőt, 1 a fekete kavicsot és 2 a fehér kavicsot.

**A képen szöveg, képernyőkép, Párhuzamos, diagram látható

Automatikusan generált leírás**

* **Modell:**
* A modell lényegi részét a Game osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint maga a játék létrehozása és a kiválasztott kavics számontartása (selectedStone). A típus lehetőséget ad új játék kezdésére (initializeMap), valamint lépésre (SlideMove). Új játéknál megadható a kiinduló játéktábla is, különben automatikusan generálódnak kezdő mezők.
* A mezők állapotváltozásáról a changedDisplay esemény tájékoztat.
* A játék végéről a gameEnd esemény tájékoztat. Az események argumentuma (isGameEndArgs) tárolja a győzelem állapotát, a győztest magát (ha van), illetve a lépések számát.
* A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)
* A képen szöveg, diagram, képernyőkép, Párhuzamos látható

  Automatikusan generált leírásA játékmódot a vezérlő gombok által átadott értékeken keresztül kezeljük és állítjuk be.
* **Nézetmodell:**
  + A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  + A nézetmodell feladatait a KitolasViewModel 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 játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.
  + A játékmező számára egy külön mezőt biztosítunk (KitolasField), amely eltárolja a kavicsok pozícióját. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).
  + A tárolt játékállapotok egy-egy StoredGameViewModel példánnyal írhatóak le. Ezek kollekcióját nem ágyazzuk be a fő nézetmodellbe (KitolasViewModel), hanem a betöltéskor és mentéskor dinamikusan állítjuk elő és adjuk át a nézet számára.

****

* **Nézet:**
* A nézetet navigációs lapok segítségével építjük fel
* A GamePage osztály tartalmazza a játéktáblát, amelyet egy Grid segítségével valósítunk meg, amelyben Button elemeket helyezünk el.
* A SettingsPage osztály tartalmazza a betöltés, mentés gombjait
* A LoadPage és a SavePage szolgál egy létező játékállapot betöltésére, illetve egy új mentésére.

* **Vezérlés:**
  + Az App osztály feladata az alkalmazás vezérlése, a rétegek példányosítása és az események feldolgozása
  + Az alkalmazás lapjait egy AppShell keretben helyezzük el. Ez az osztály felelős a lapok közötti navigációk megvalósításáért

A képen szöveg, képernyőkép, levél, dokumentum látható

Automatikusan generált leírás**Tesztelés:**

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a KitolasGameModelTest osztályban.
* Az alábbi tesztesetek kerültek megvalósításra:
  + TestContructor: Új játék indítása, és a SlideTable objektum adattagjainak helyességének ellenőrzése.
  + TestMatrixStonePlaces: A mátrixban a kavicsok helyének ellenőrzése.
  + TestSlide: A SlideTable slide metódusának helyességének ellenőrzése valid illetve invalid lépésekkel.
  + TestInvalidSteps: A slide metódus nem megfelelő (ha a játékos lelépne a tábláról) lépéseinek ellenőrzése.
  + TestChangeSelectedStone: A program kavics léptető funkcionalitásást vizsgáló teszt.
  + TestGameEndings: A program játék végét ellenőrző metódúsát és eseményét ellenőrző teszt.