# Feladatleírás

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy 𝑛 × 𝑛 mezőből álló tábla, amelyen két játékos űrhajói helyezkednek el, középen pedig egy fekete lyuk. A játékos 𝑛 − 1 űrhajóval rendelkezik, amelyek átlóban helyezkednek el a táblán (az azonos színűek egymás mellett, ugyanazon az oldalon). A játékosok felváltva léphetnek. Az űrhajók vízszintesen, illetve függőlegesen mozoghatnak a táblán, de a fekete lyuk megzavarja a navigációjukat, így nem egy mezőt lépnek, hanem egészen addig haladnak a megadott irányba, amíg a tábla széle, a fekete lyuk, vagy egy másik, előtte lévő űrhajó meg nem állítja őket (tehát másik űrhajót átlépni nem lehet). Az a játékos győz, akinek sikerül űrhajóinak felét eljuttatnia a fekete lyukba. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (5 × 5, 7 × 7, 9 × 9), valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

# Feladat elemzése

* A játéktáblának 3 mérete van, kicsi: 5x5, közepes: 7x7, nagy: 9x9. Mindegyik pályához n-1 számú hajó tartozik / játékos, ahol n a tábla mérete. Inicializálás után a játék alapértelmezetten kis méretű pályát tölt be.
* A feladat egy .NET MAUI alkalmazásként fogjuk megvalósítani, elsődlegesen Windows és Android platformokra.
* Az ablak tetején öt gomb fog helyet foglalni az alábbi funkciókkal:
  + Új játék: ennek 3 variációja lesz, 1-1 minden játékmérethez
  + Játék mentése: erre kattintva a felhasználó elmentheti egy szöveges fájlba a játék jelenlegi állását
  + Játék betöltése: a felhasználó kiválaszthat egy mentésfájlt, amiből be lehet tölteni az abban foglalt információk alapján egy játékállást.
* A táblákat gombokból álló rácsként fogjuk megjeleníteni, melyek mindegyikére egészen a játék végéig rá lehet kattintani. Amennyiben egy hajóra kattint a játékos akkor kiválasztja az említett hajót, amely után megjelennek az „irányítómezők”, melyekkel mozgathatja a kiválasztott hajót. Amennyiben ki van választva egy hajó és a játékos egy másikra kattint, az új hajó veszi át a kiválasztott szerepét, valamint, ha ki van választva egy hajó és félrekattintunk, akkor visszavonjuk a kiválasztást. Egy játékos mindig csak a saját hajójára tud kattintani, és sose lesz egyszerre több hajó kiválasztva.

Amint az egyik játékos nyert, lezárjuk a gombrácsot, onnantól nem lehet majd folytatni a játékot, és a mentés opciót is letiltjuk.

# Felhasználói esetdiagram

Játékos

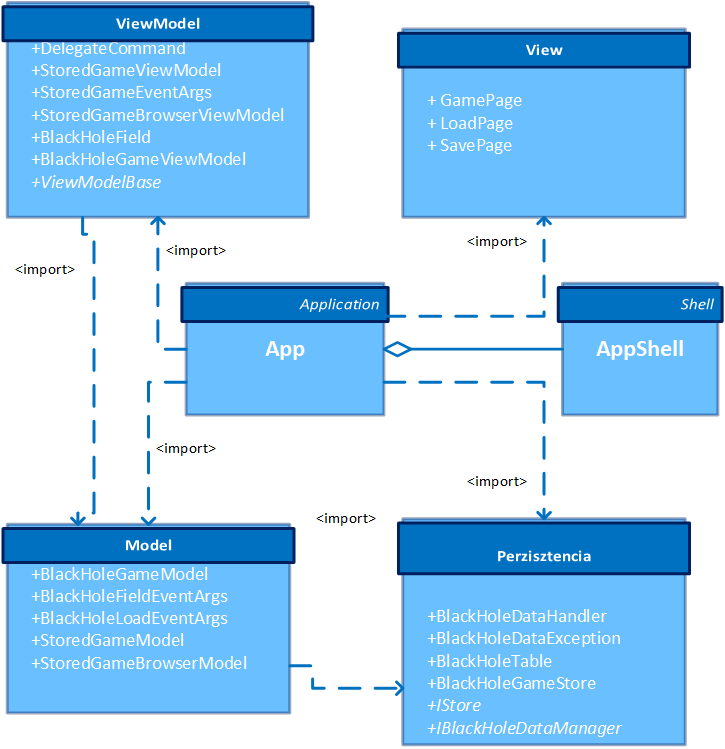
<<precedes>>

<<precedes>>

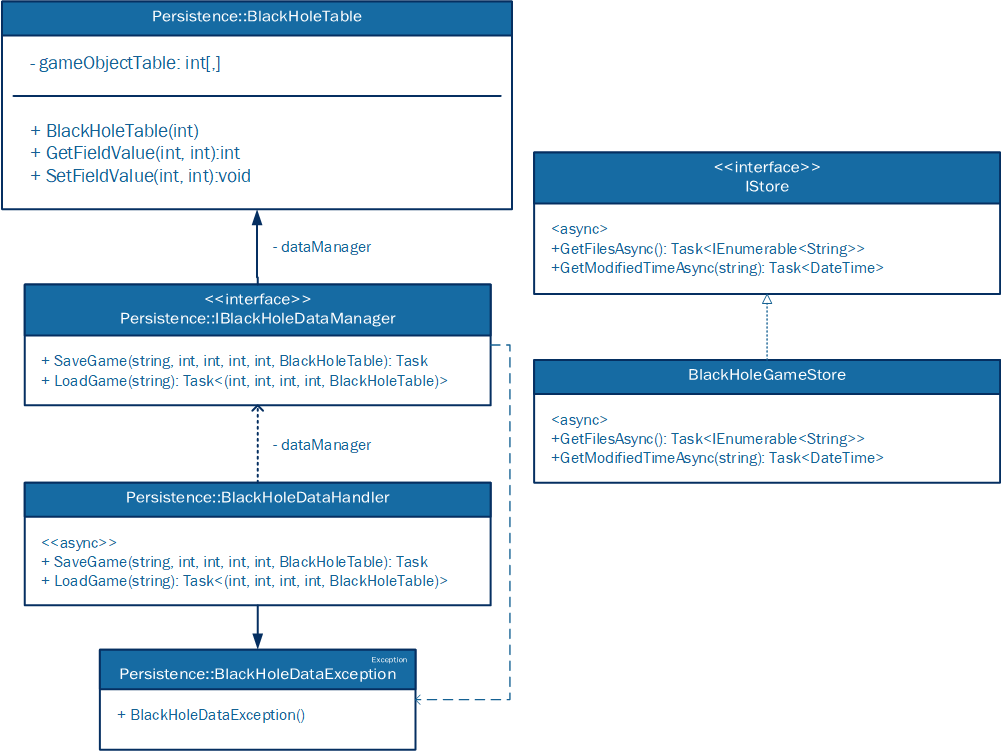
<<invokes>>

# Tervezés

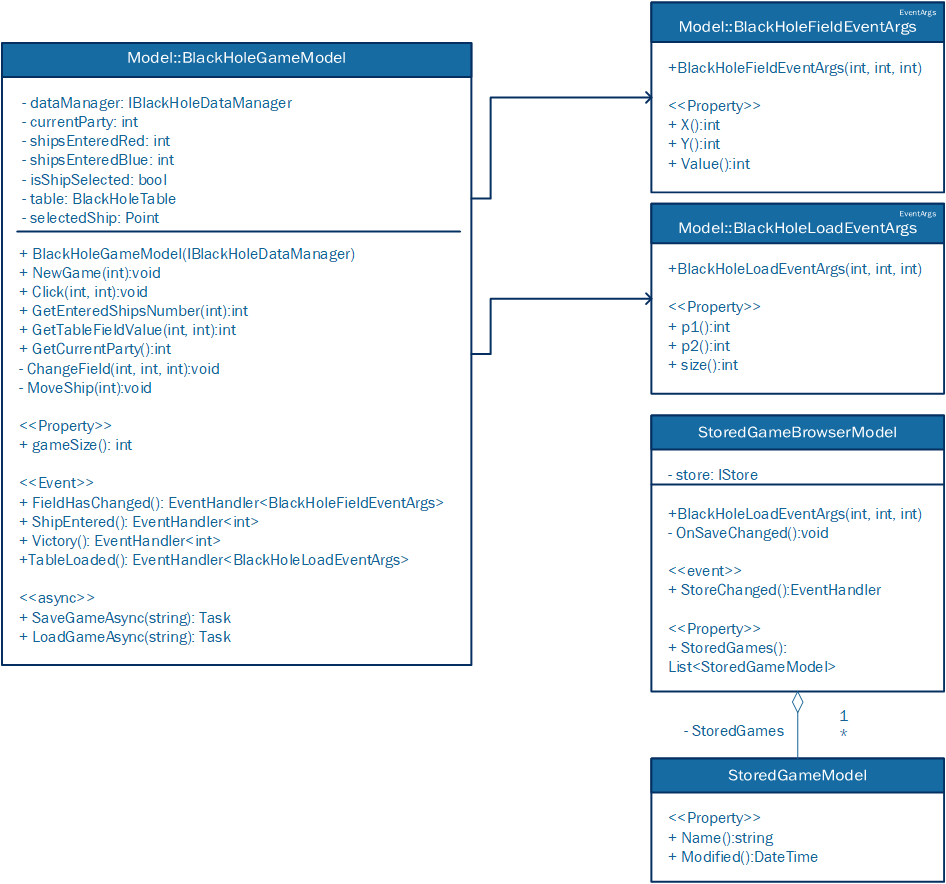
## Programszerkezet:

* A programot MVVM architektúrában valósíjuk meg, ennek megfelelően View, ViewModel, Model és Persistence névtereket hozunk létre az alkalmazáson belül.
* A megvalósításból külön építjük fel a játék, betöltés és mentés funkciókat valamennyi rétegben.
* A program vezérlését az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodellt és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.

## Perzisztencia:

* Az adatkezelés feladata a játéktáblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
* A BlackHoleTable osztály egy játéktáblát biztosít, ahol minden mezőnek ismert az értéke (0-üres mező, 1-piros játékos hajója, 2-kék játékos hajója, 3-fekete lyuk). A tábla alapértelmezés szerint 5 × 5 -ös, de ez a konstruktorban paraméterezhető. A tábla lehetőséget ad az értékek lekérdezésére (GetFieldValue) és módosítására (SetFieldValue).
* A hosszú távú adattárolás lehetőségeit az IBlackHoleDataManager interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadGame), valamint mentésére (SaveGame). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
* Az interfészt szöveges fájl alapú adatkezelésre a BlackHoleDataHandler osztály valósítja meg. A fájlkezelés során fellépő hibákat a BlackHoleDataException kivétel jelzi.
* A program az adatokat szöveges fájlként tudja eltárolni, melyek a bht kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást – amennyiben a játék nem lezárult állapotban van.
* A fájl első sora megadja az aktív játékos számát, a második sorában pedig a két játékos elért pontszámát kapjuk meg külön-külön. A harmadik sorában megadjuk, hogy mekkora a tárolt tábla mérete (továbbiakban n). A fájl többi része izomorf leképezése a játéktáblának, azaz összesen n sor következik, és minden sor n számot tartalmaz szóközökkel választva. A számok 0-3 közöttiek lehetnek, ahol 0 reprezentálja a még üres mezőt.

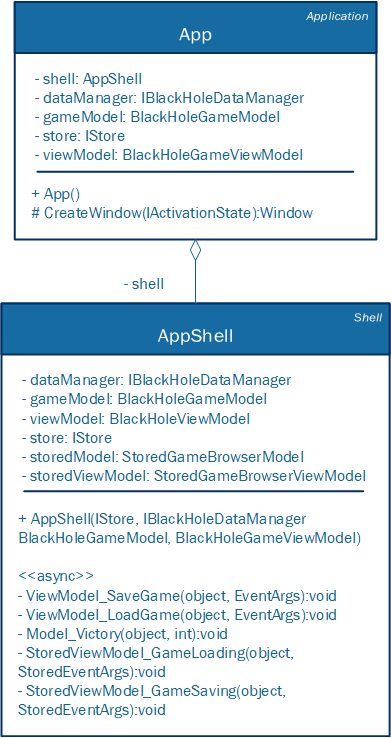
## Modell

* A modell lényegi részét a BlackHoleGameModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, például az egyes játékosok megszerzett pontjait (shipsEnteredRed, shipsEnteredBlue), vagy az éppen soron következő játékost (currentParty). A típus lehetőséget ad új játék kezdésére (NewGame), valamint a kattintások lekezelésére (Click), amikor is a klikkelt mező értékétől függően fog műveletet végrehajtani. Új játéknál mérettől függően automatikusan generálódik le egy alapértelmezett tábla.
* A mezők állapotváltozásáról a FieldHasChanged esemény tájékoztat. Az esemény argumentuma (BlackHoleFieldEventArgs) tárolja a megváltozott mező pozícióját és étékét. Az új szerzett pontokról a ShipEntered esemény, míg a játék végéről a Victory esemény tájékoztat. Az utóbbi események argumentuma (int) tárolja a győztes fél számát, illetve az új pontot szerző játékos számát.
* A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGame) és mentésre (SaveGame).

## Nézet

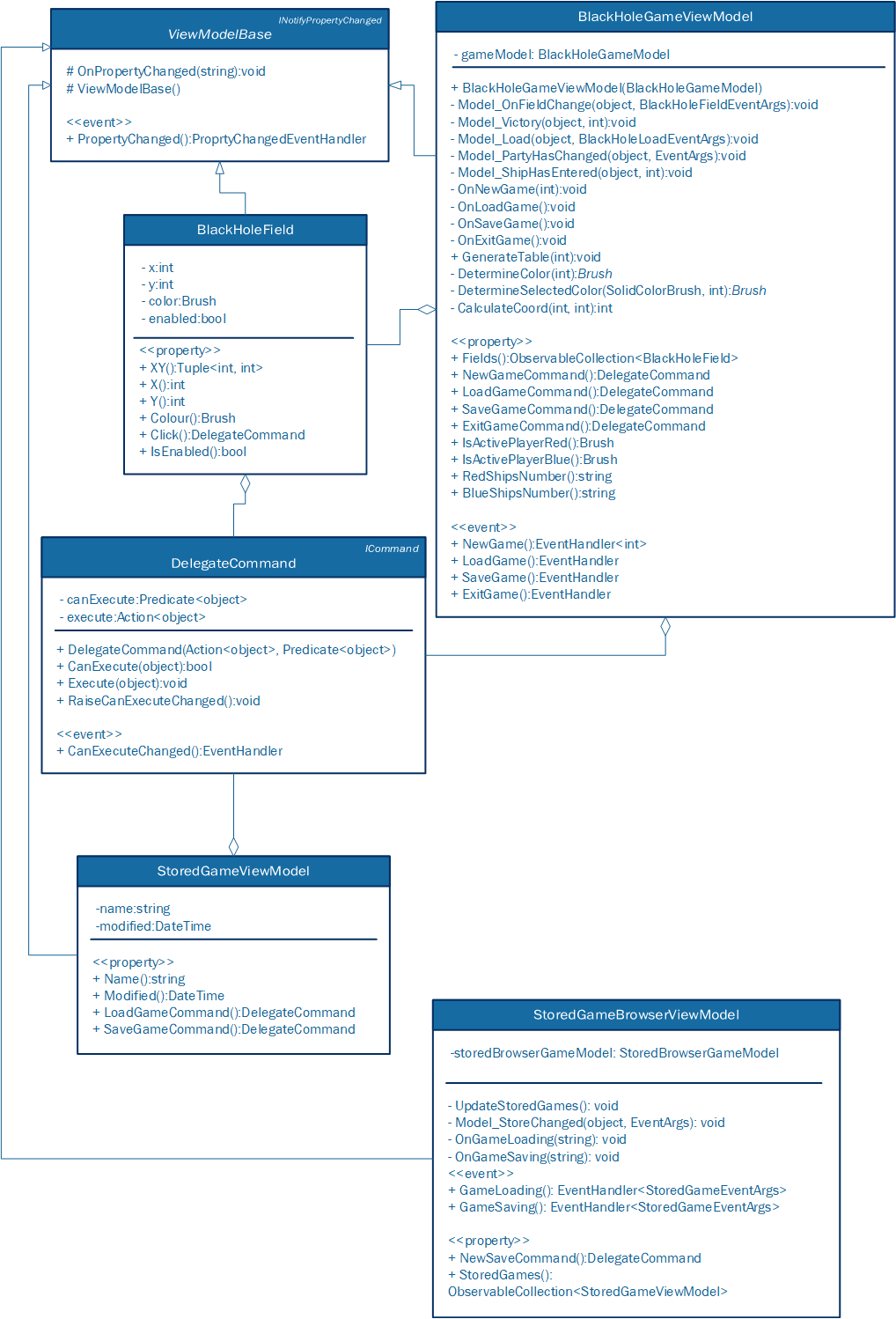
* A nézet csak egy képernyőt tartalmaz, a MainWindow osztályt. A nézet egy rácsban tárolja a játékmezőt, a menüt és a státuszsort. A játékmező egy ItemsControl vezérlő, ahol dinamikusan felépítünk egy rácsot (UniformGrid), amely gombokból áll. Minden adatot adatkötéssel kapcsolunk a felülethez, így a gombok színét, az aktív játékos jelzését és a sikeresen bevezetett hajók számát.

## Vezérlés, környezet

* Az App osztály feladata az egyes rétegek példányosítása (App\_Startup), összekötése, a nézetmodell, valamint a modell eseményeinek lekezelése, és ezáltal a játék, az adatkezelés, valamint a nézetek szabályozása.

## Nézetmodell

* A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
* A nézetmodell feladatait a BlackHoleGameViewModel 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 (gameModel), de csupán információkat kér le tőle. Direkt nem avatkozik a játék futtatásába.
* A játékmező számára egy külön mezőt biztosítunk (BlackHoleField), amely eltárolja a pozíciót, színt, engedélyezettséget, valamint a lépés parancsát (Click). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).



# Tesztelés

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a BlackHoleGameModelTest osztályban.
* Az alábbi tesztesetek kerültek megvalósításra:
  + BlackHole\_Game\_NewGame\_Test
    - Generálunk egy új táblát, és a helyességének ellenőrzéséhez megszámoljuk a legenerált hajókat az egyik félnek, valamint ellenőrizzük, hogy a pontok az egyes játékosoknak 0-ra vannak állítva
  + BlackHole\_Game\_LoadGame\_Test
    - Egy előre beállított táblát töltünk be mockolt perzisztencia réteg segítségével, és a betöltött tábla egyes mezőinek specifikus vizsgálatával ellenőrizzük a betöltés sikerességét
  + BlackHole\_Game\_StepTaken\_Test
    - Az LoadGame teszt sikerességére építve betöltjük az előre generált táblánkat, majd a modell Click metódusát meghívva szimuláljuk egy hajó léptetését, ami után ellenőrizzük a tábla mezőinek helyességét.
  + BlackHole\_Game\_ShipEntered\_Test
    - Újfent betöltjük az előre beállított táblánkat, ami lehetőséget ad egy léptetéssel a pontszerzésre. A léptetés előtt ellenőrizzük a felek pontszámát, majd a léptetés végrehajtása után ellenőrizzük, hogy a megfelelő fél kapta-e a pontot
  + BlackHole\_Game\_Victory\_Test
    - Újfent betöltjük az előre beállított táblánkat, ami lehetőséget ad egy léptetéssel a pontszerzésre, ezzel pedig a játék megnyerésére. A lépés előtt létrehozunk kettő változót, amelyekkel azt fogjuk ellenőrizni, hogy megnyertük-e a játékot, illetve, hogy melyik fél nyerte meg. Ezeket a modell Victory eventjéhez rendelt metódussal fogjuk megváltoztatni, és a nyertes lépés végrehajtása után ellenőrizzük, hogy megváltoztak-e a változóink értékei