

3. Eva Beadandó feladat

Készítette:

Név: Klenovszky Áron

Neptun kód: H9MRBV

Email: h9mrbv@inf.elte.hu

Feladat - 10. Aknamező:

Készítsünk programot a következő játékra.

A játékban egy tengeralattjárót kell irányítanunk a képernyőn (balra, jobbra, fel, illetve le), amely felett ellenséges hajók köröznek, és folyamatosan aknákat dobnak a tengerbe. Az aknáknak három típusa van (könnyű, közepes, nehéz), amely meghatározza, hogy milyen gyorsan süllyednek a vízben (minél nehezebb, annál gyorsabban).

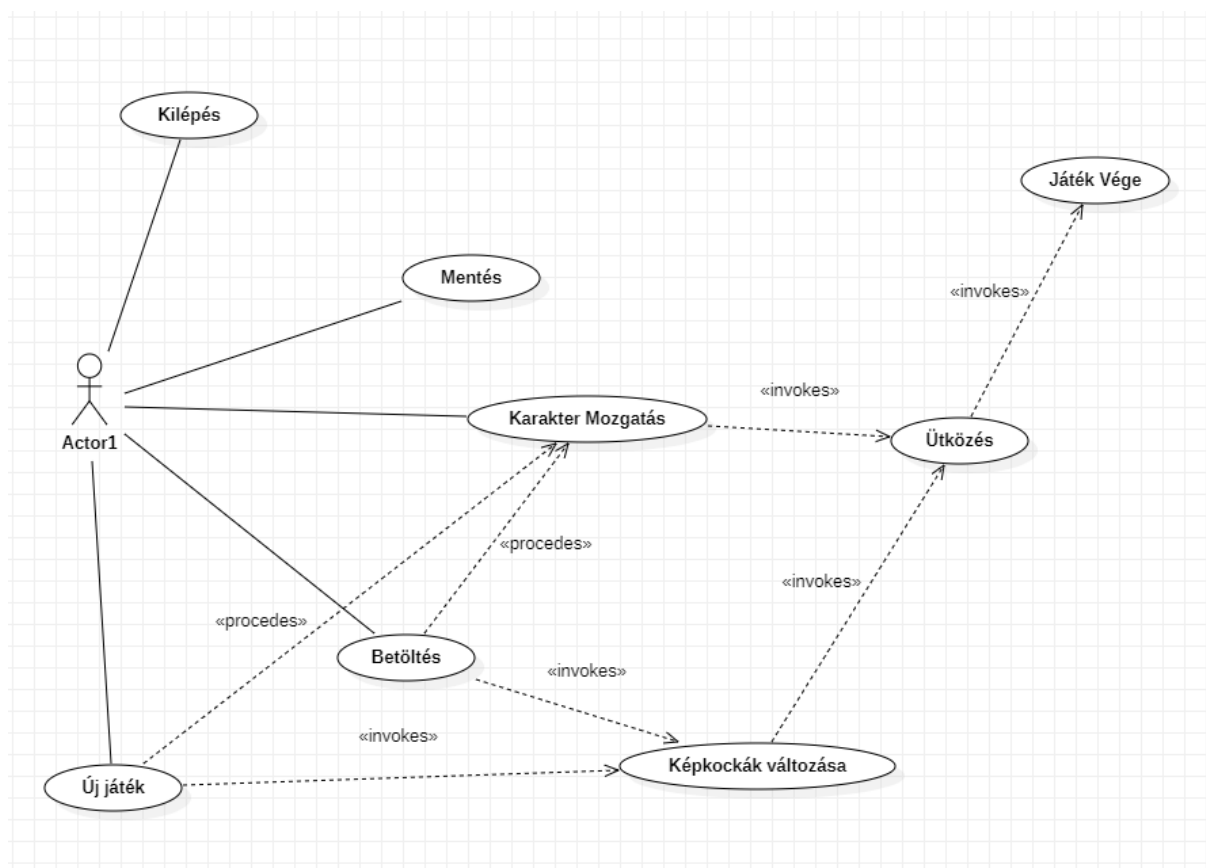
Az aknákat véletlenszerűen dobják a tengerbe, ám mivel a hajóskapitányok egyre türelmetlenebbek, egyre gyorsabban kerül egyre több akna a vízbe. A játékos célja az, hogy minél tovább elkerülje az aknákat. A játék addig tart, ameddig a tengeralattjárót el nem találta egy akna.

A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére.

Elemzés:

- A feladatot .NET MAUI alkalmazásként, elsődlegesen Windows és Android platformon valósítjuk meg, kiegészítve a beépített DisplayAlert-ek által generált dialógusablakkal.
- A felső sorba egy menü helyezkedik el, egy a következő gombokkal:
 - *New Game* - új játékot kezd
 - *Load Game* - betölt egy elmentett játékot
 - *Save Game* - elmenti az aktuális játékállást
 - *Pause* - Leállítja a játékot
- Az alsó sorban a játék státusza látható:
 - *Game Time: 00:00:00* - az aktuális játék ideje

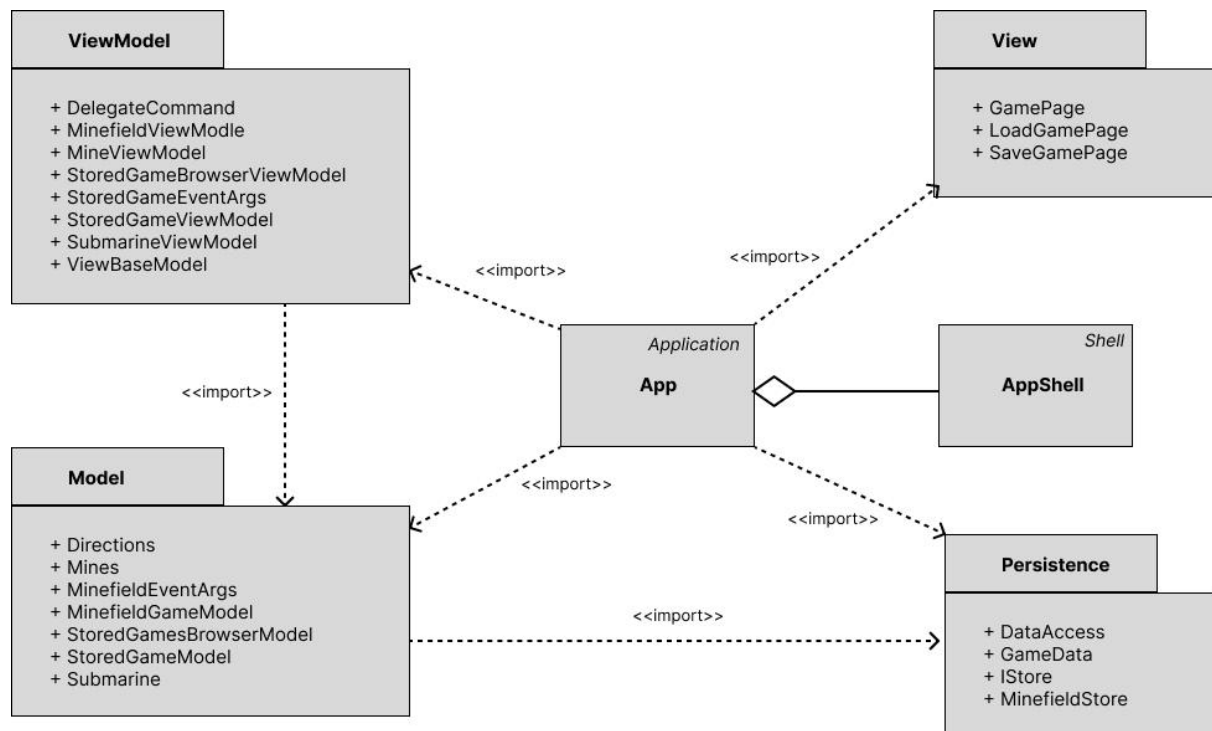
- *Paused* - a *Pause* gomb lenyomásával meg lehet állítani az aktuális játékot, akkor ez a felirat jelenik meg
- A játék a 600 x 900 pixeles, fix méretű ablak szinte teljes méretében zajlik a fej és a lábléc kivételével. A játékos a billentyűzeten található nyilakkal irányíthatja a tengeralattjárót. A bombák az ablak tetejéről 3 különböző sebességgel haladnak lefelé.
- Ha a játékos karaktere érintkezik egy bombával, akkor egy dialógusablak jelzi ezt a játékos számára, majd új játék kezdhető.
- A mentés és a kilépést szintén a beépített dialógusablak oldja meg. A menteni kívánt file nevét a játékos adja meg, és ő is választja meg a betölti kívánt file-t.
- A felhasználói esetek az 1. ábrán láthatóak.



1. ábra : Felhasználói esetek diagramja

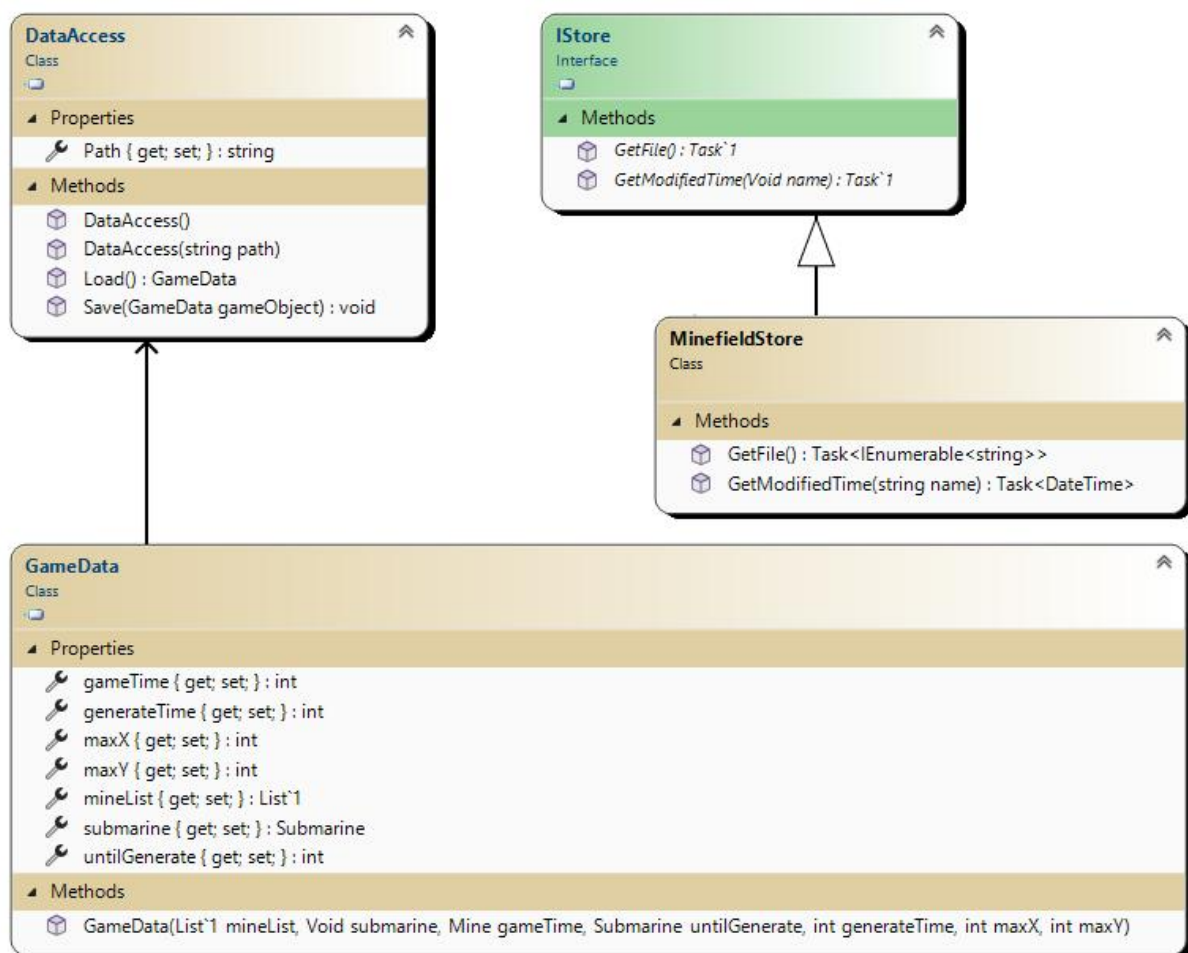
Tervezés:

- Programszerkezet:
 - A program szerkezete 3 projektre van szétosztva.
 - Minefield - *Class Library*
 - Minefield.MAUI - *MAUI App*
 - MinefieldTest - *xUnit Test Project*
 - MVVM architektúraában valósul meg a program:
 - View - megjelenítés (Minefield.MAUI)
 - ViewModel - környezet (Minefield.MAUI)
 - Model - modellek (Minefield)
 - Persistence - perzisztencia (Minefield)
 - A program vezérlését az App és az AppShell adja
 - A csomagszerkezet a 2. ábrán látható



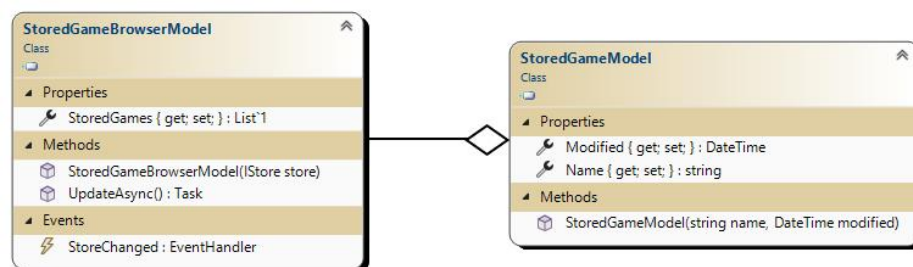
2. Ábra: Csomagdiagram

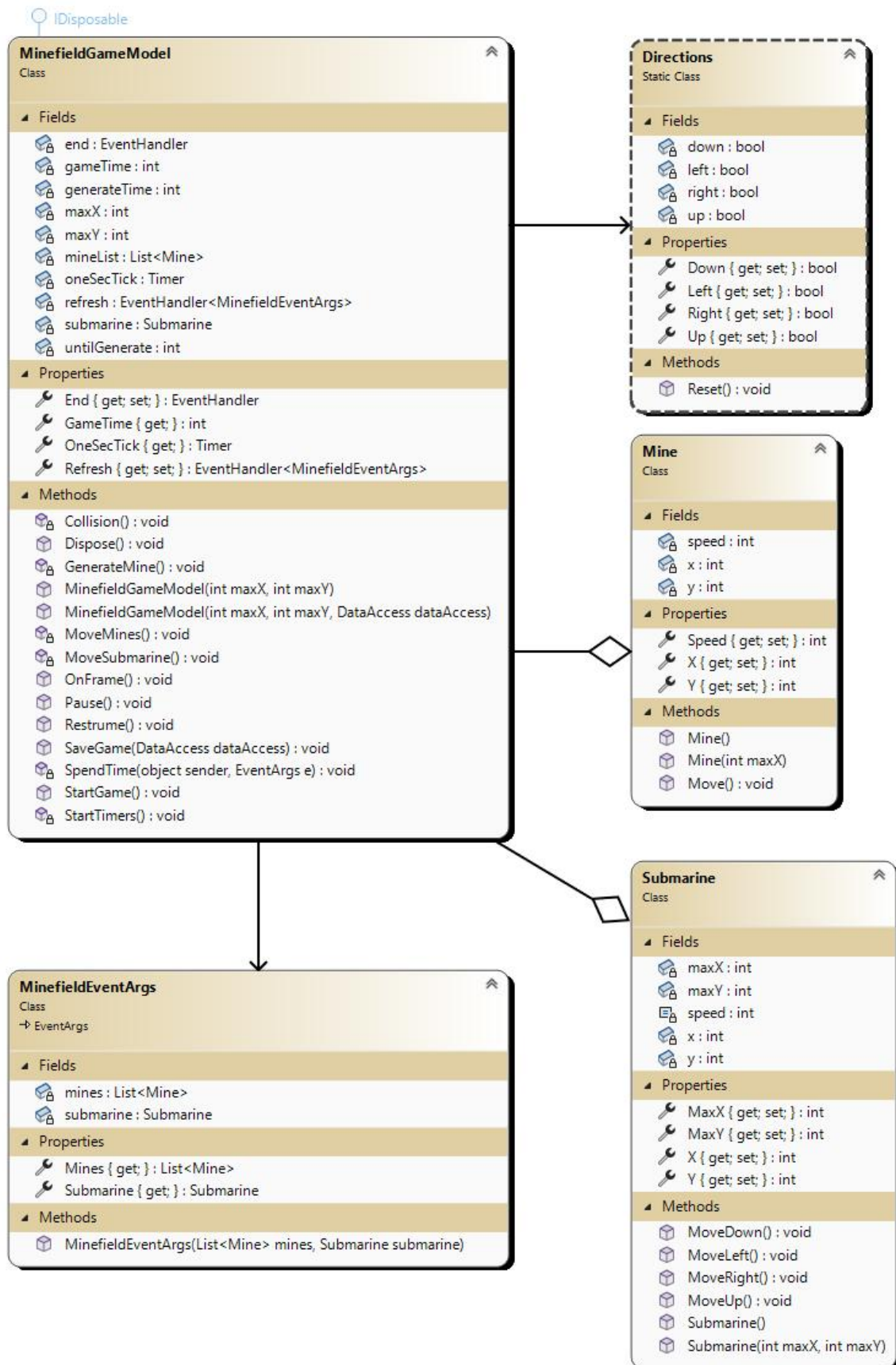
- Perzisztencia (3. ábra):
 - Feladata az adatkezelés, a játék adatainak betöltése és mentése.
 - A GameData egy olyan osztály, amely tárolja egy játékhoz tartozó összes információt. A játékból eddig eltelt idő, a játékelemek pozíciója, valamint a bomba generáló adott állapotát tartalmazza. Akkor kerül létrehozásra, amikor a játék mentődik, vagy betöltődik. Ehhez fér hozzá a DataAccess a játék tárolásával kapcsolatban.
 - A DataAccess felel a JSON típusú állományok létrehozásáért egy GameData objektumból mentés (Save()) során, és a JSON file beolvasásáért egy GameData objektumba a betöltés (Load()) során. Hiba esetén egy hibát dob a felhasználó felé.
 - A mentett JSON file teljes mértékben a GameData osztályt valósítja meg annak minden alosztályával együtt.
 - A MinefieldStore lekérdezi a választott mappában található fájlokat, és egyesével a hozzájuk tartozó utolsó módosítási dátumot.



3. ábra: A Persistence csomag osztálydiagramja

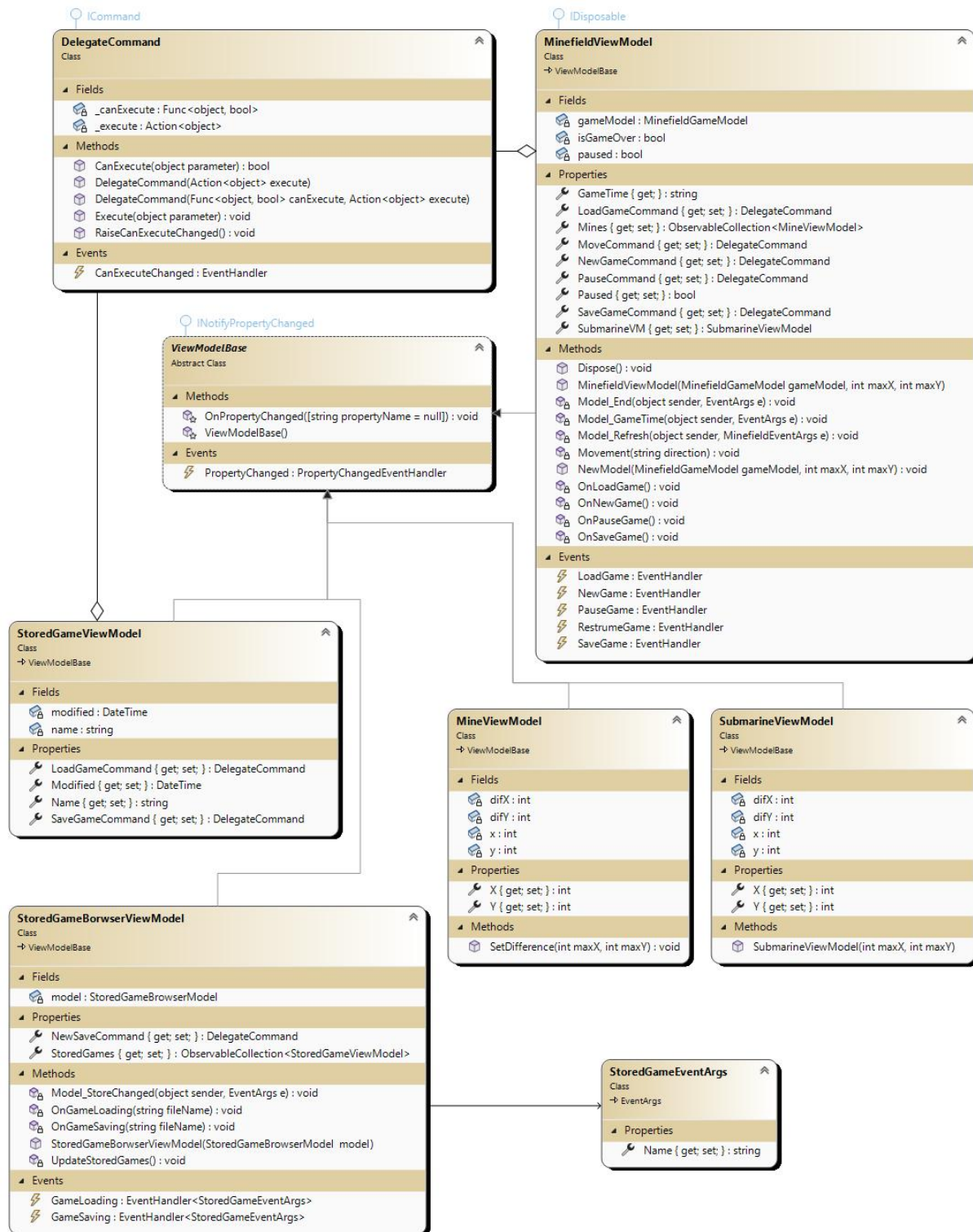
- Modell (4. ábra):
 - A Submarine osztály tartalmazza a játékos karakterének mechanikai szempontból vett összes adatát (sebesség, koordináták, mozgáshatárok). Ezen kívül megvalósítja a karakter mozgását mind a négy irányba, és ellenőrzi azokat.
 - A Mine osztály tartalmazza egy bomba adatait (koordináták, sebesség), melyeket a konstruktora hívásának esetén véletlenszerűen generál a megadott kereteken belül. Valamint a Move() függvény segítségével mozgatja is azt a sebességének megfelelően.
 - A Directions egy statikus osztály, ami tárolja az éppen aktivált irányokat a hajó mozgásához.
 - A MinefieldEventArgs az EventArgs osztályt megvalósító osztály, ami a játék aktuális állását - a tengeralattjáró és a bombák helyzetét - közvetíti a View felé.
 - A MinefieldGameModel felelős a játék lényegi részének megvalósításáért. Tárolja az aktuális játékelemeket, és vezérli azokat.
 - Felel a játék elindításáért, megállításáért, folytatásáért.
 - Az OnFrame() függvény mozgatja (és generálja) a bombákat és a tengeralattjárót. Értesíti a View-t, ha frissíteni kell a játékkeret. Ellenőrzi hogy a bombák ütköznek e a játékos karakterével, és ha igen, akkor értesíti erről a View-t, hogy le lehessen állítani a játékot.
 - Számolja az eltelt időt.
 - Kétféle konstruktorral rendelkezik. Hogy elindítson egy új játékot, szüksége van az ablak méretére. Ha pedig egy játékot akar betölteni a felhasználó, akkor megkapja a már példányosított adatkezelő DataAccess-t, és annak segítségével feltölti az adatait.
 - A mentés esetében a példányosított DataAccess-t a SaveGame() függvénye kapja meg, amelynek átadja a menteni kívánt adatokat egy GameData objektum formájában.
 - A StoredGameBrowserModel és a StoredGameModel (lásd alább) felelnek a mentett játékok adatainak tárolásáért és frissítéséért.





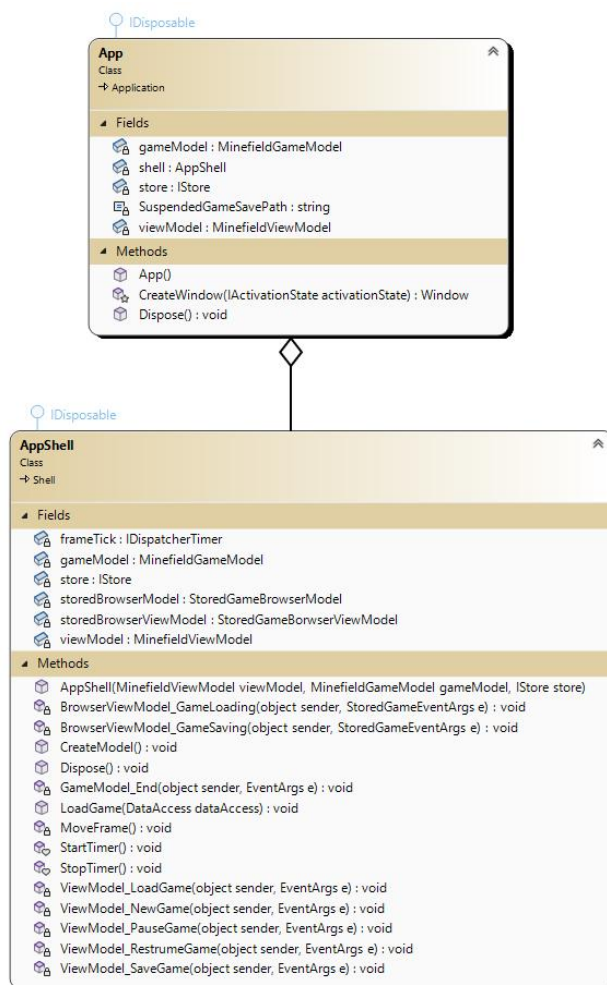
4. ábra: A Model csomag osztálydiagramja

- Nézetmodell (5. ábra)
 - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
 - A **MinefieldViewModel** osztály látja el a nézetmodell feladatait.
 - Biztosítja a parancsokat a játék kezdéséhez, betöltéséhez, mentéséhez és kilépéshez.
 - A parancsokhoz kapcsolt események jeleznek a környezetnek.
 - Tárolja a MinefieldGameModel egy hivatkozását, hogy abból adatokat kérjen le, annak eseményei révén.
 - A játék objektumai számára a MineViewModel és SubmarineViewModel osztályokat használja pozíciójuk tárolására.
 - Az objektumok view modelljeiben lévő pozíciókat az ablak méretéhez képest torzítják, hogy a megfelelő helyeken jelenjenek meg.
 - A tárolt játékállapotok egy-egy StoredGameViewModel példánnyal írhatóak le. Ezek kollekcióját nem ágyazzuk be a fő nézetmodellbe, hanem a betöltéskor és mentéskor dinamikusan állítjuk elő és adjuk át a nézet számára.



5. ábra: A nézetmodell osztálydiagramja

- Nézet:
 - A nézet navigációs lapok segítségével épül fel.
 - A GamePage osztály tartalmazza a játék felületét, melyben a pixelek pozíciójához igazodnak a játék objektumai.
 - A SavePage és a LoadPage osztály szolgál a játék állásának mentésére, illetve a mentett játékok közül az elindítani kívánt betöltésére.
- 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.



6. ábra: A vezérlés osztálydiagramja

Tesztelés:

- A modellek ellenőrzése egységtesztetek segítségével lett megvalósítva a MinefieldTest névtér alatt, xUnite használatával.
 - *SubmarineTest* - A tengeralattjáró létrehozásának, mozgásának és annak korlátainak tesztelése.
 - *MineTest* - A bombák létrehozásának és mozgásának tesztelése.
 - *MinefieldGameModelTest*
 - NewGameTest - Új, alap állásról induló játék, és az idő számláló működésének tesztelése.
 - OnFrameTest - Bombák és a Tengeralattjáró egy "Tick" alatt való elmozdulásának helyességének és a Játék végének elkerülésére irányuló teszt.
 - LoadGameTest - A modell már mentett játékból való betöltésének, és egy léptetésének tesztelése mock-olt perzisztencia réteggel.
 - *GameModelEnd* - A játék végének a tesztelése ütközés esetén.
 - *GenerateMine* - A játék folyamán bombák létrejöttének tesztelése.
 - *MineFallOut* - A bombák eltűnésének a tesztelése, ha azok kiértek a képernyőről.
 - *PauseContinue* - A játék megállításának, valamint annak újraindításának tesztelés.