2.Beadandó feladat

Készítette:

Csóka Máté
E-mail: gt9bq9@inf.elte.hu

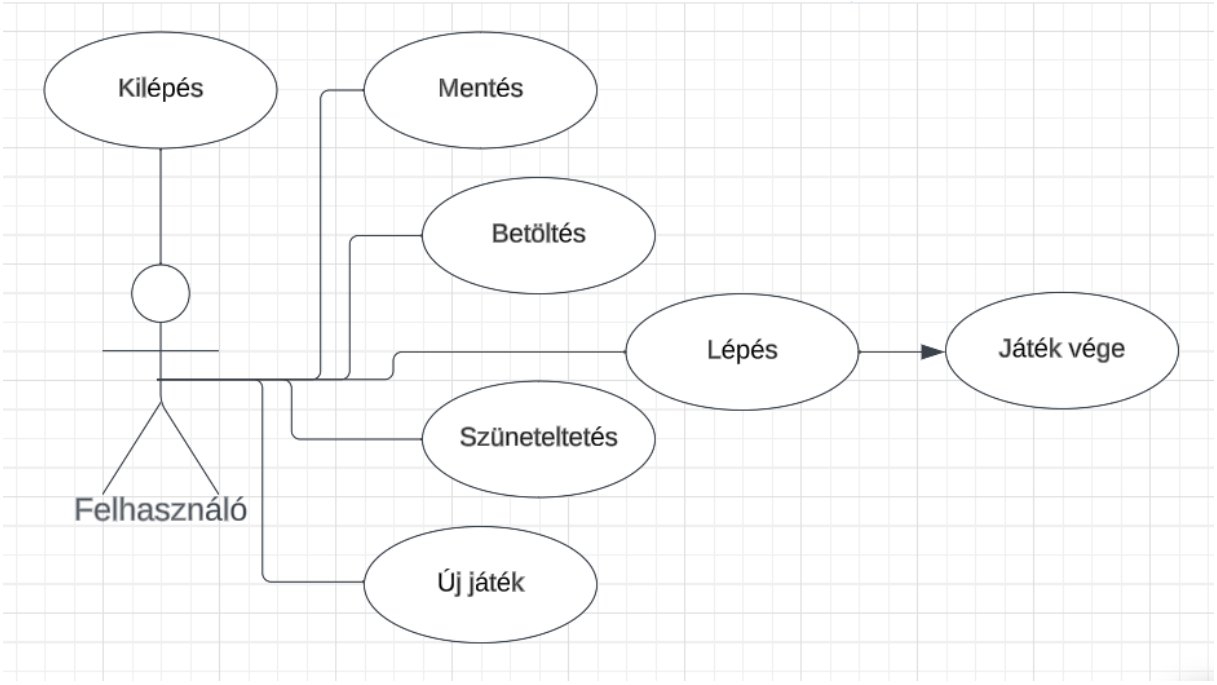
Feladat:

Készítsünk programot, amellyel az alábbi motoros játékot játszhatjuk. A feladatunk, hogy egy gyorsuló motorral minél tovább tudjunk haladni. A gyorsuláshoz a motor üzemanyagot fogyaszt, egyre többet. Adott egy kezdeti mennyiség, amelyet a játék során üzemanyagcellák felvételével tudunk növelni. A motorral a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg üzemanyagcellák, amelyek folyamatosan közelednek a képernyő alja felé. Mivel a motor gyorsul, ezért a cellák egyre gyorsabban fognak közeledni, és mivel a motor oldalazó sebessége nem változik, idővel egyre nehezebb lesz felvenni őket, így egyszer biztosan kifogyunk üzemanyagból. A játék célja az, hogy a kifogyás minél később következzen be. 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. Az alkalmazás három lapból fog állni. Az alkalmazás portré tájolást támogat.
- 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 harmadik 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.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: New Game, Start, Pause, Save, Load. Az ablak alján megjelenítünk egy státuszsort, amely az üzemanyagszintet, illetve a hátralévő időt jelzi.
- A játéktáblát egy 13x13 gombokból álló rács reprezentálja. A rács alsó sorában van a motor (feketével jelzett mező), ezt irányíthatja a játékos a nyilak megnyomásával. A rács felső sorában véletlenszerűen jelennek meg üzemanyagcellák (pirossal jelzett mező), egyszerre csak egy és erre is csak 30% esély van. Az üzemanyagcellák egyre jobban gyorsulva esnek lefelé. Az üzemanyagszint folyamatosan csökken, az idő

múlásával egyre gyorsabban. Ha az üzemanyagcella, érintkezik a motor cellával, akkor az üzemanyagszinthez, +5 egység adódik. Maximum üzemanyagszint 50 egység lehet.
- A játék automatikusan kidob egy dialógusablakot, amikor vége a játéknak (elfogyott az üzemanyag). Szintén dialógusablakkal végezzük el a mentés, betöltést.
- A felhasználói esetek az 1. ábrán láthatóak.

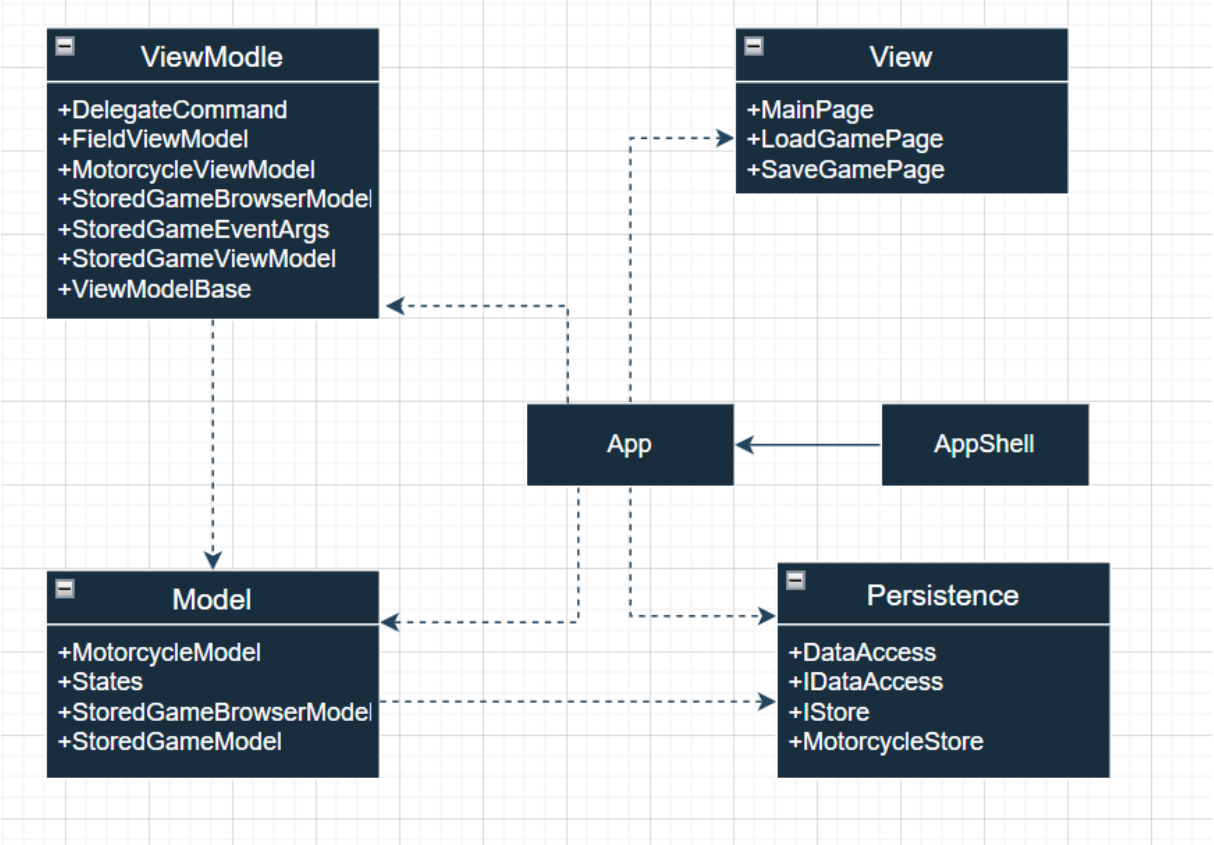


1.ábra: Felhasználói esetek diagramja

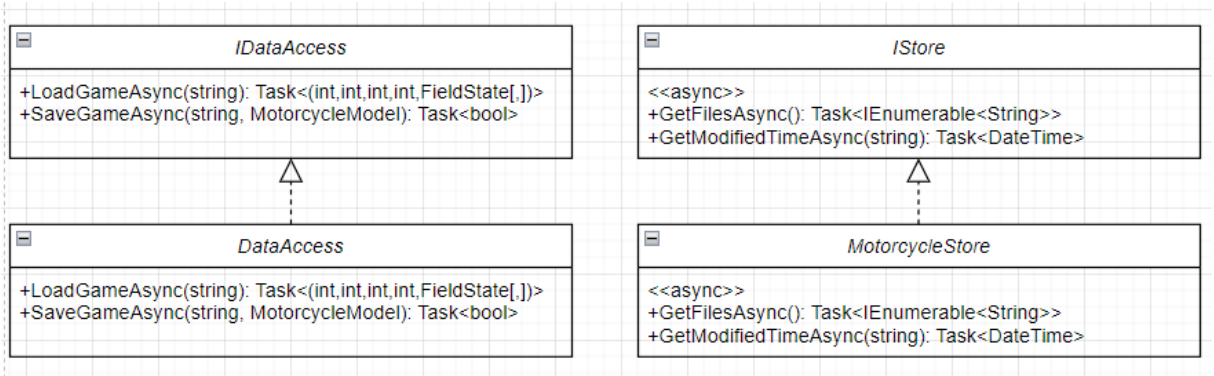
Tervezés:

- 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 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 ***Persistence*** és ***Model*** csomagok felületfüggetlenek, míg a ***ViewModel*** és ***View*** csomagok WPF függőek. Egy projekten belül valósítottam meg őket.
- Perzisztencia (3. ábra):
  - Az adatkezelés feladata a Motorcycle táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  - A hosszú távú adattárolás lehetőségeit az ***IDataAccess*** interfész adja meg, amely lehetőséget ad a tábla betöltésére (***LoadGame***), valamint mentésére (***SaveGame***).

- Az interfész szöveges fájl alapú adatkezelésre a DataAccess osztály valósítja meg.
- A program az adatokat szöveges fájlként tudja eltárolni. 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 második sor az időt, majd sorban következik: az üzemanyagszint, a sebesség, majd a mátrix értékei.



2.ábra: Az alkalmazás csomagdiagramja

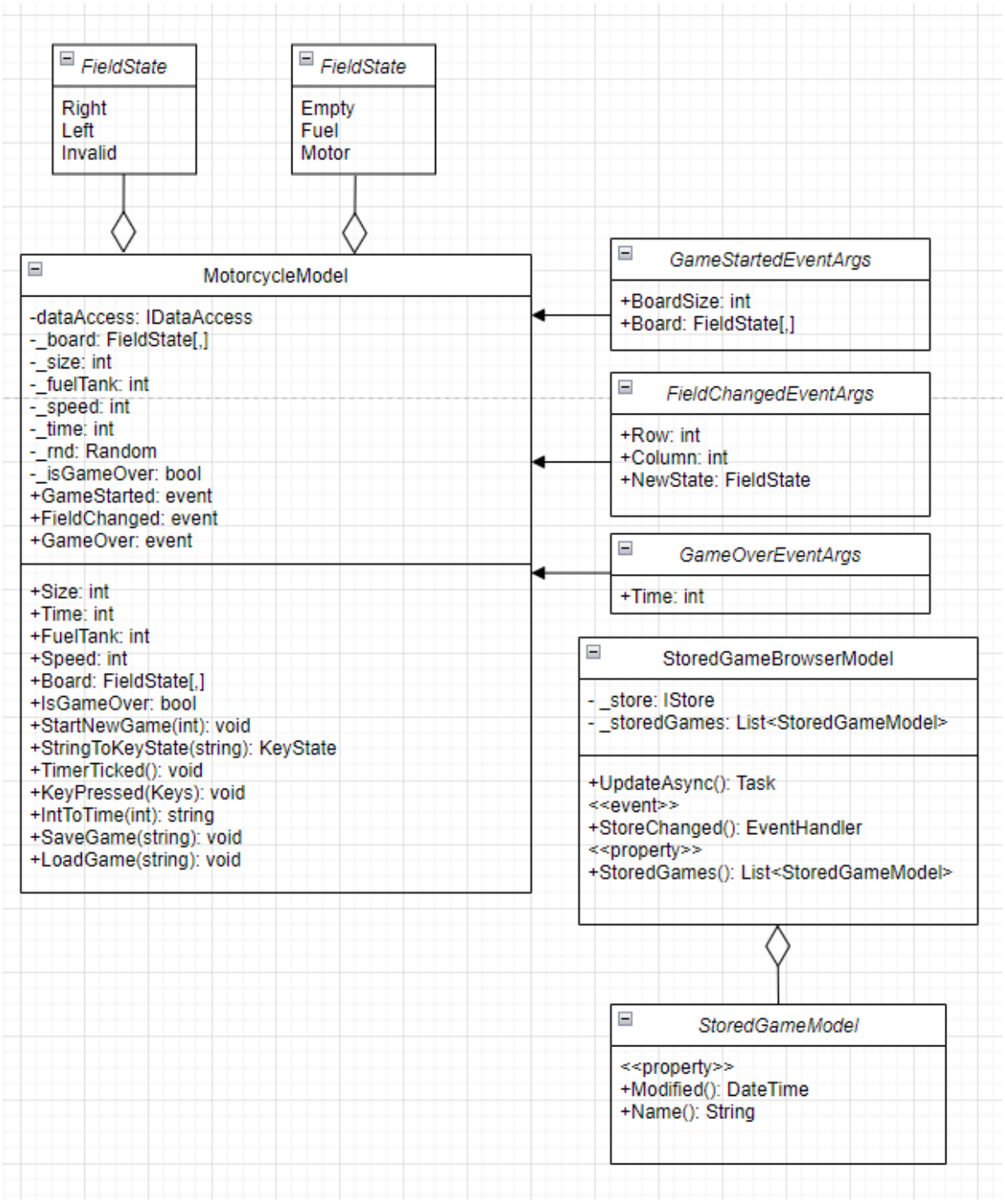


3.ábra: A Perzisztencia osztály csomagdiagramja

- Modell (4. ábra):
  - A modell lényegi részét a ***MotorcycleModel*** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit,

úgymint az idő (***TimerTicked***). A típus lehetőséget ad új játék kezdésére (***startNewGame***), valamint lépésre (KeyPressed).

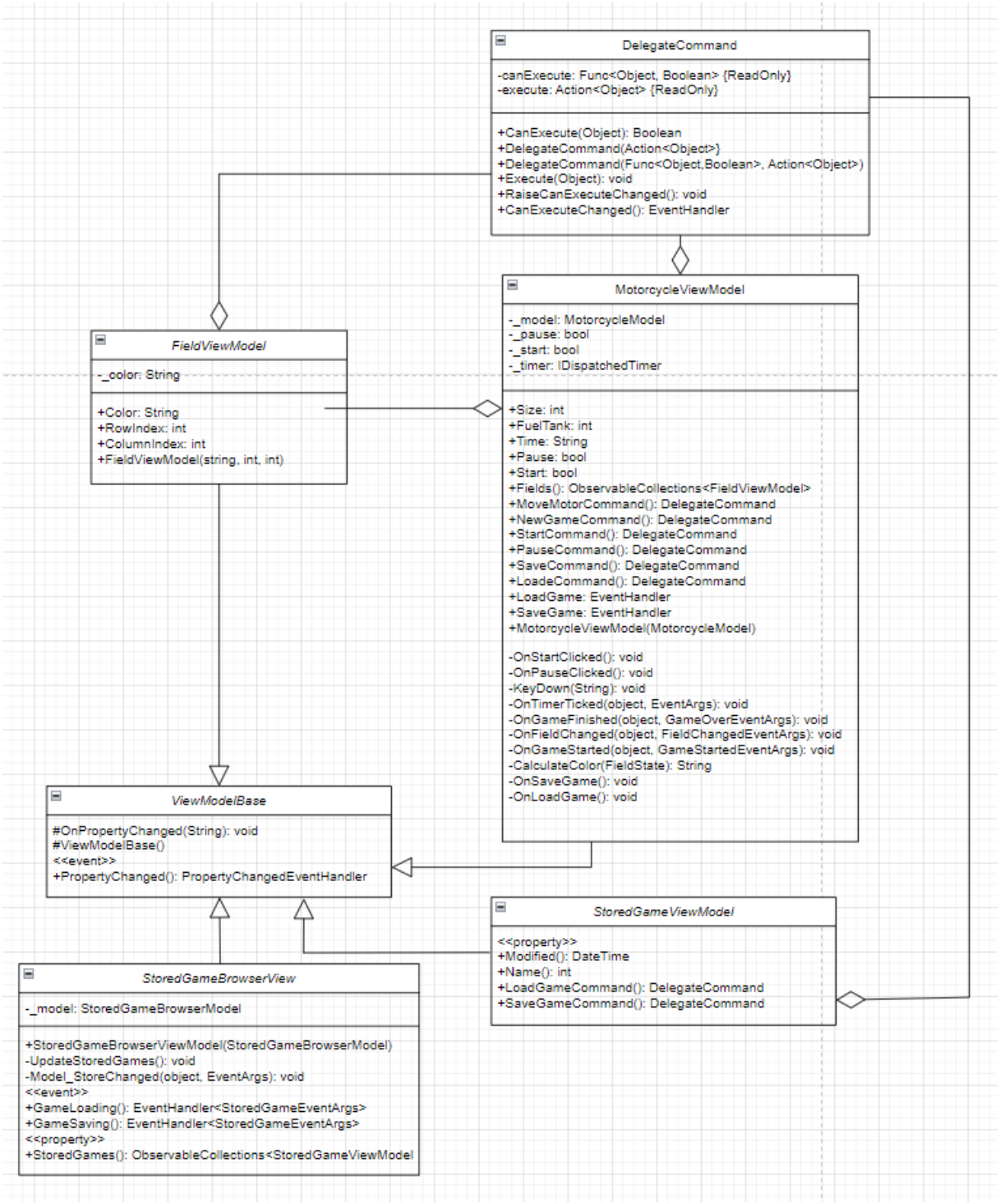
- A játékállapot változásáról a ***FieldChangedEventArg*** esemény, míg a játék végéről a ***GameOverEventArg*** esemény tájékoztat.
- A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (***LoadGame***) és mentésre (***SaveGame***)



4.ábra: A Modell csomag osztálydiagramja

- Nézetmodell (5. ábra):

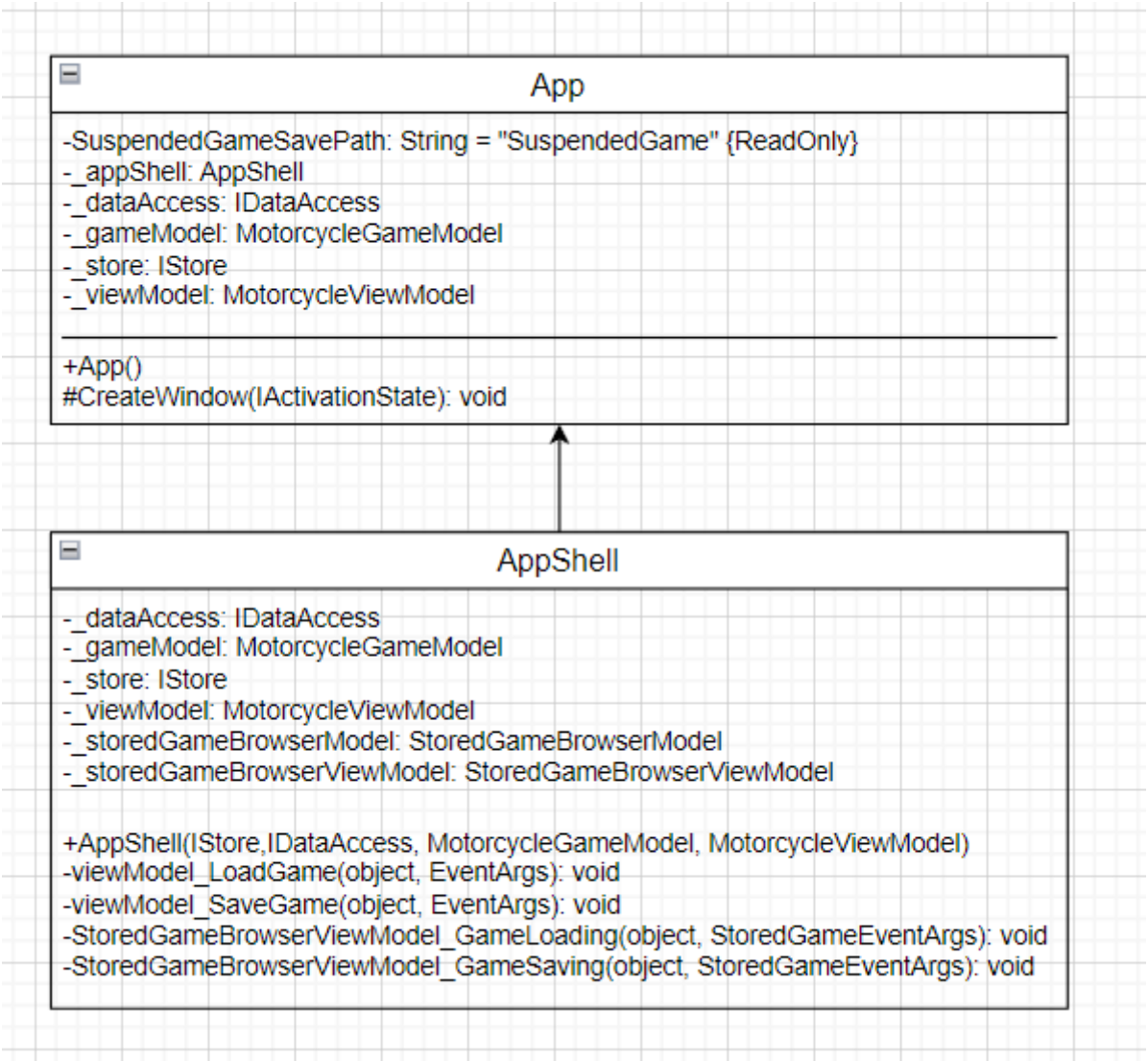
- A nézetmodell megvalósításához felhasználunk egy általános utasítás (***DelegateCommand***), valamint egy ős változásjelző (***ViewModelBase***) osztályt.
- A nézetmodell feladatait a ***MotorcycleViewModel*** osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, 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 (***FieldViewModel***), amely eltárolja a pozíciót és a cellák színét. 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 (**MotorcycleViewModel**), hanem a betöltéskor és mentéskor dinamikusan állítjuk elő és adjuk át a nézet számára.



5. ábra: A nézetmodell osztálydiagramja

- Nézet:
  - A nézetet navigációs lapok segítségével építjük fel.
  - A **MainPage** osztály tartalmazza a játéktáblát, amelyet egy **Grid** segítségével valósítunk meg, amelyben **Rectangle** elemeket helyezünk el.
  - 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.



6. ábra: A vezérlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **MotorcycleGameTest** osztályban.
- Az alábbi tesztek kerültek megvalósításra:

- **MotorcycleModelNewGameTest**: Új játék indítása, a mezők kitöltése, valamint a gyorsaság, üzemanyagmennyiség és az eltelt idő ellenőrzése.
- **MotorcycleModelStepRightTest, MotorcycleModelStepLeftTest**: Játékbeli lépések hatásainak ellenőrzése.
- **MotorcycleModelTickedTest:** Játékbeli időtelés hatásainak ellenőrzése. (Üzemanyagcella lejjebb esése, gyorsulás, idő megnövelése)
- **MotorcycleModelCatchedFuelCellTest:** A motor által elkapott üzemanyagcellák hatásainak ellenőrzése. (Üzemanyagtank megfelelő növelése)
- **MotorcycleModelLoadTest:** A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.