**3. csoport Dokumentáció**

**Készítette:**

Ács Botond – DF9FL6 – acs.botond42@gmail.com

Dóra László – IO89I5 – doralaci98@gmail.com

Jakab Olivér – CAGJCQ – jakaboliver98@gmail.com

**Ismertetés**:

A feladat egy robotrendszer kialakítása, ami egy raktár logisztikáját automatizálja, mely precíz és gyors feladatvégzést tud végezni folyamatosan. A robotok közvetlen a polcok alá mennek, melyet felemelnek és az egész polcot viszik a célállomáshoz. Így a raktárnak szükséges rendelkezni a robotokhoz tartozó energia-töltőállomással, illetve célállomással ahová a robot a polcokat viszik közvetlen, melyekről az alkalmazottak a megfelelő termékeket leveszik további csomagolásra. Ilyenkor a robot visszaviszi a polcot az eredeti helyére.

A program számolja a robotok elhasznált energiáját, a leszállított csomagok darabszámát. A programnak támogatnia kell mint a beolvasott szimulációs adatokkal való szimuláció újraindítását illetve egy új adatokkal való új szimuláció kialakítását.

Ezzel fel lehet gyorsítani a raktári logisztikát, hosszú távon pénzt lehet spórolni a munkaerőn, illetve precíz állandó munkavégzésre lehet számítani.

**Mérföldkövek:**

1. 7. hétre alap prototípus és teljes git használat.
2. 10. hétre 90%-os usercase teljesítmény, unit testek.
3. 13. hétre végső termék.

Ezeken a nagyobb mérföldköveken kívül a fejlesztési munkák folyamatosan zajlanak. Hetente minimum egy élőben / internetes platformon összeülés, folyamatok megbeszélése, munkák további kiosztása.

**Elemzés**:

* A feladatot egyablakos asztali alkalmazásként *Windows Presentation Foundation* grafikus felülettel valósítjuk meg.
* A program indításakor, fájlbeolvasással olvasunk be egy előre megadott NxM alakú táblát.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új szimuláció, Jelenlegi szimuláció újraindítása, Kilépés), Sugó.
* Az ablak alján megjelenik egy státuszsor, ami számolja a lépésszámot, leadott termékeke és az összes energiafogyasztást.
* A szimulációs táblát egy NxM-es címkékből álló rács reprezentálja.
  + Üres mező – fehér
  + Robot – Sárga
  + Polc – Szürke
  + Töltőállomás – Kék
  + Leadási állomás – Zöld
  + Robot polcot alatt – Sötétebb narancssárga
  + Robot viszi a polcot – Piros
* A felhasználói esetek az első ábrán láthatóak.

tervezett

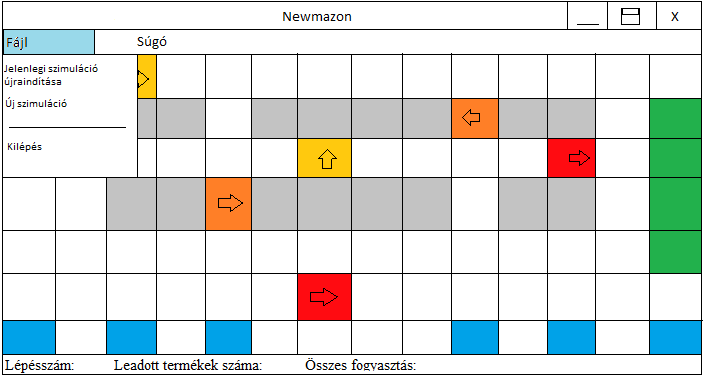
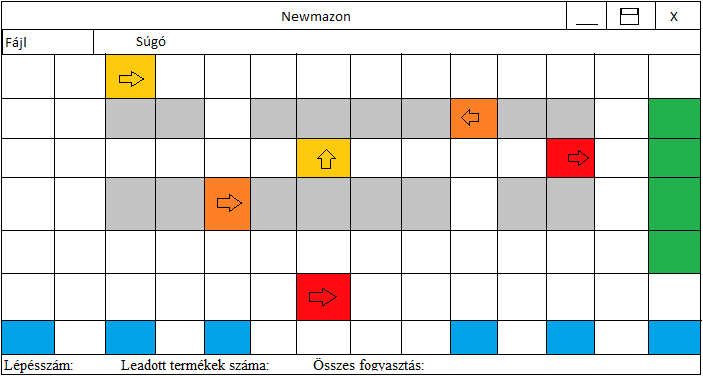
Felhasználó

Felhasználói esetek

**Nemfunkcionális követelmények:**

* A program biztonságos, hisz egy központi vezérlő egység vezérli a robotokat, mely folyamatosan frissíti azok helyeit.
* A program válaszideje a robotok útkereső algoritmusával megegyező.
* Maga a program tárfoglalása minimális.
* A program egyszerűen hordozható és skálázható.
* Jogi korlátok egyenlőre nincsenek, mivel iskolai project és nem forgalmazható.

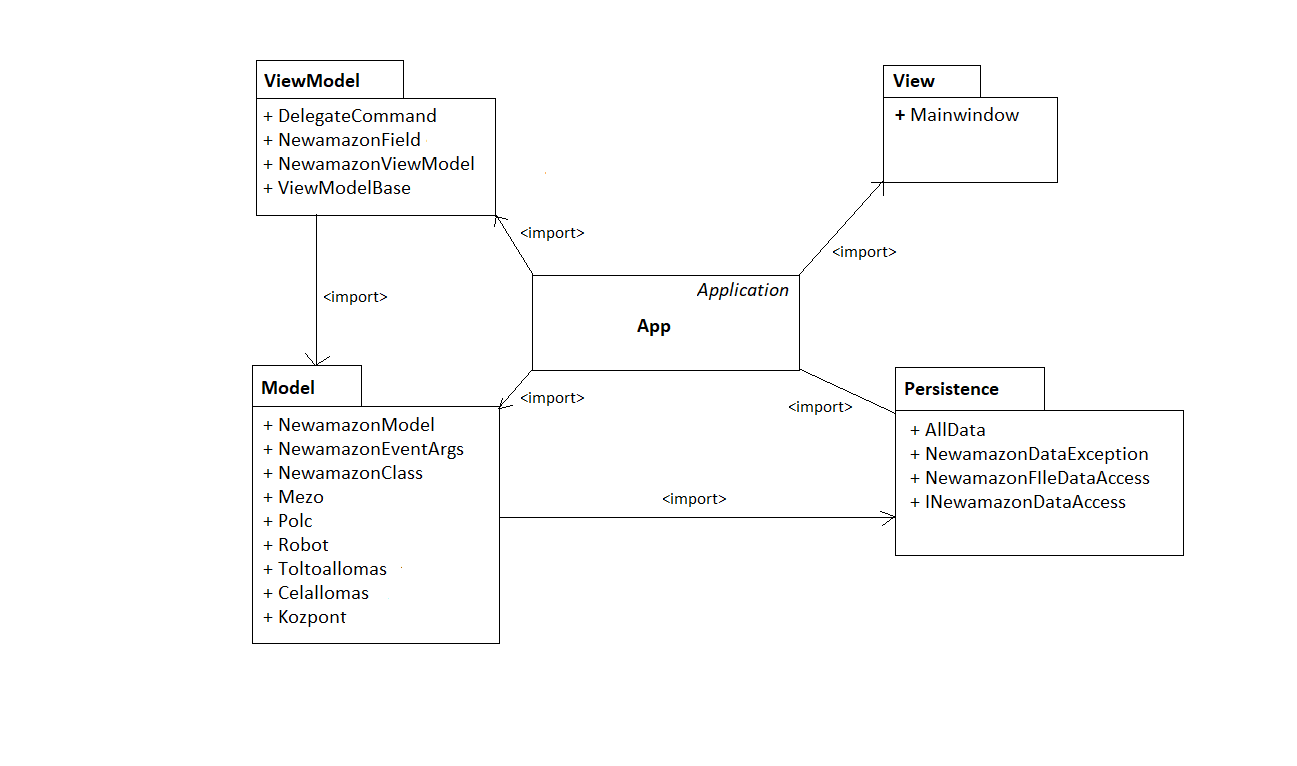
**Felhasználói felület:**



**Felhasználói történetek (még nem végleges)**

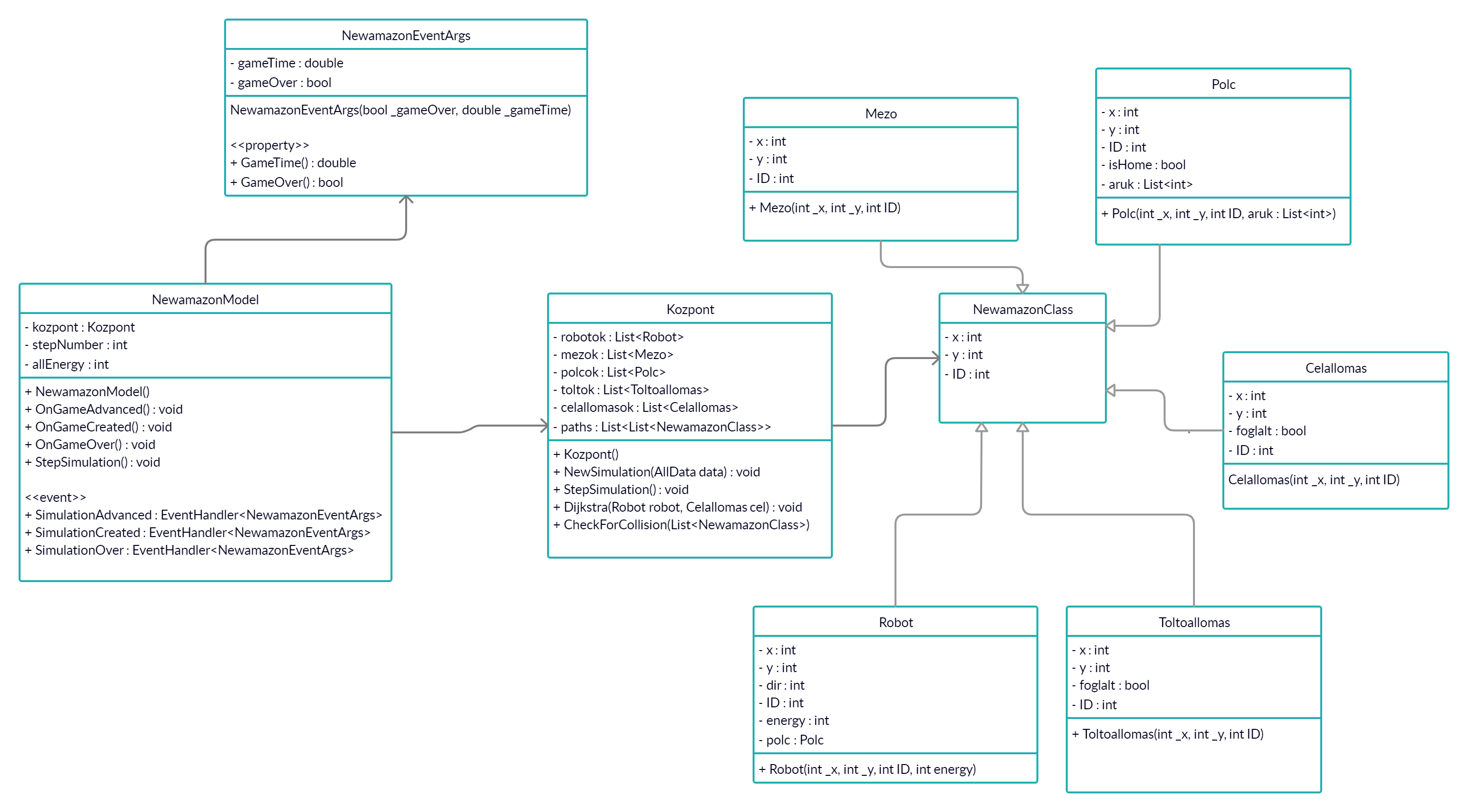
Felhasználóként szeretném, hogy:

* fájlbeolvasással tudjak választani előre megírt alapállapotok között azért, hogy többféle szimuláció működését tudjam ellenőrizni.
* a robotok időben fel tudják magukat tölteni a lehető legközelebbi töltőállomáson azért, hogy ne merüljenek le menet közben.
* a robotok egy polc alatt tartózkodva meg tudják emelni azt azért, hogy utána el tudják vinni a célállomásra.
* a robotok tudjanak a polcok alatt közlekedni (ha éppen nem cipelnek polcot) a könnyebb közlekedés érdekében.
* a robotok valamilyen módon ki tudják kerülni egymást azért, hogy fennakadásmentesen működjön a szimuláció. (nem fő prioritás, csak ha már minden mással kész vagyunk)

**Osztályok áttekintése**

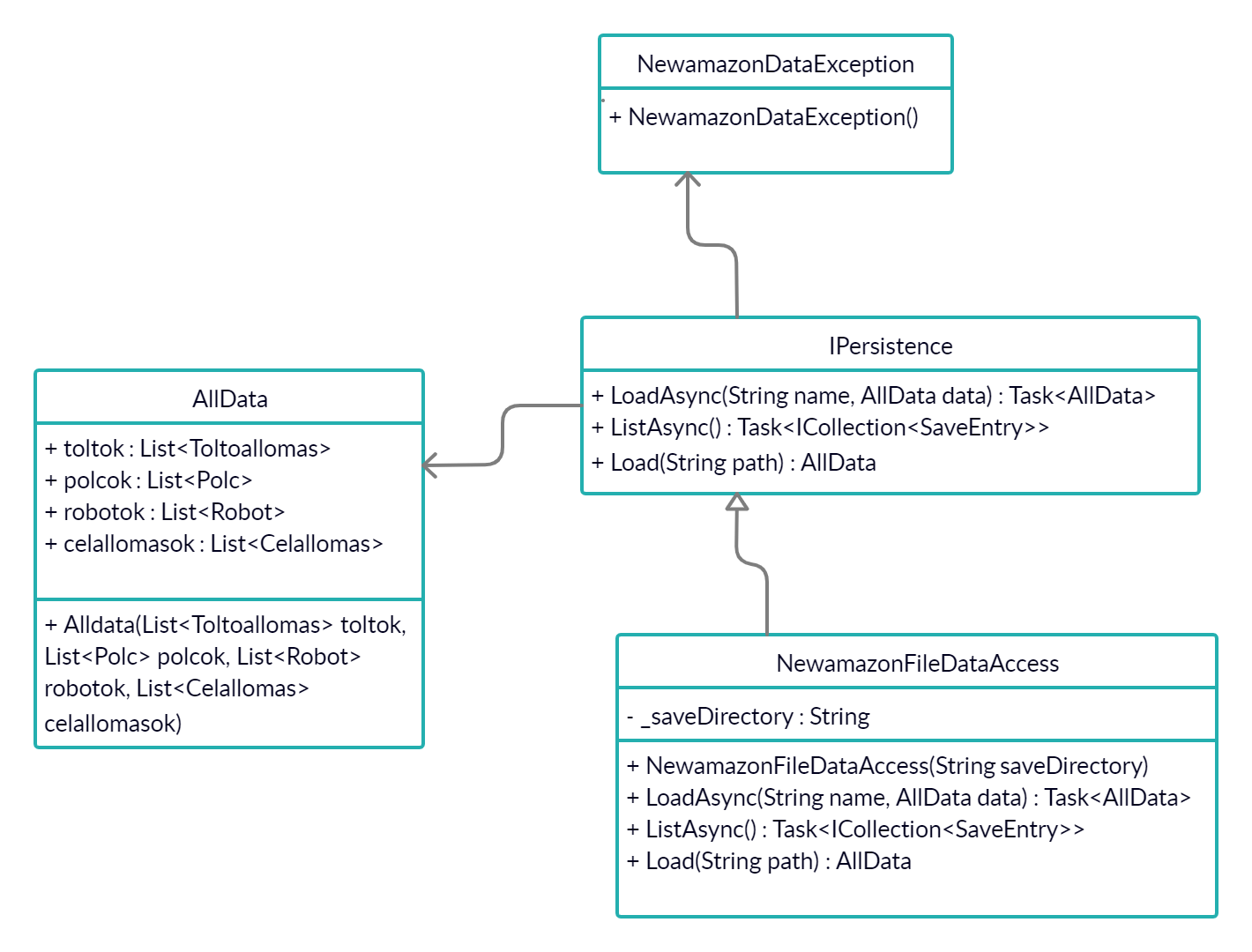
**Model:**

* NewamazonModel.cs
* NewamazonEventArgs.cs
* NewamazonClass.cs
* Mezo.cs
* Polc.cs
* Robot.cs
* Toltoallomas.cs
* Celallomas.cs
* Kozpont.cs



**Persistance:**

* AllData.cs
* NewamazonDataException.cs
* NewamazonFileDataAccess.cs
* INewamazonDataAccess.cs

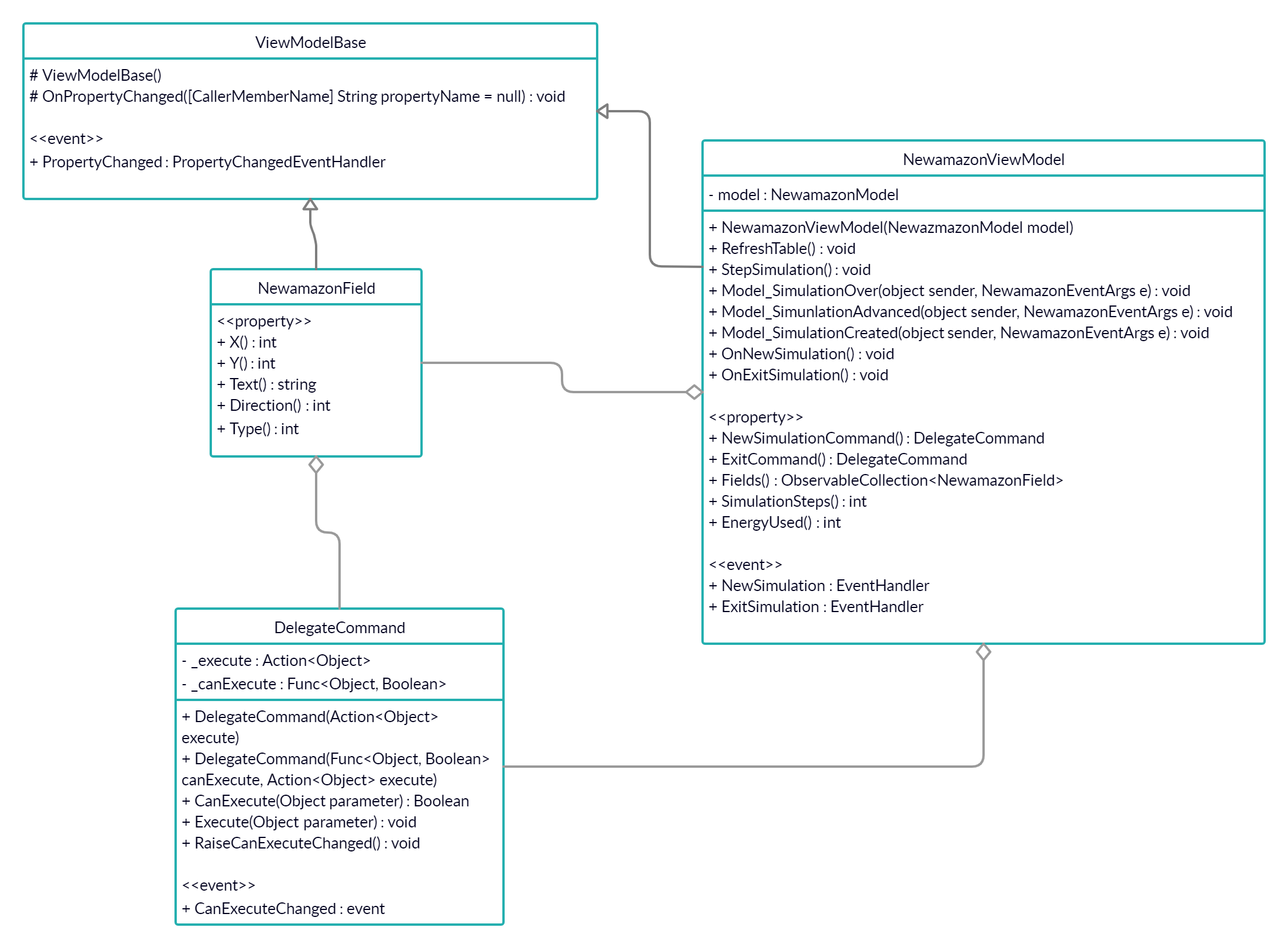


**View:**

* MainWindow.xaml.cs

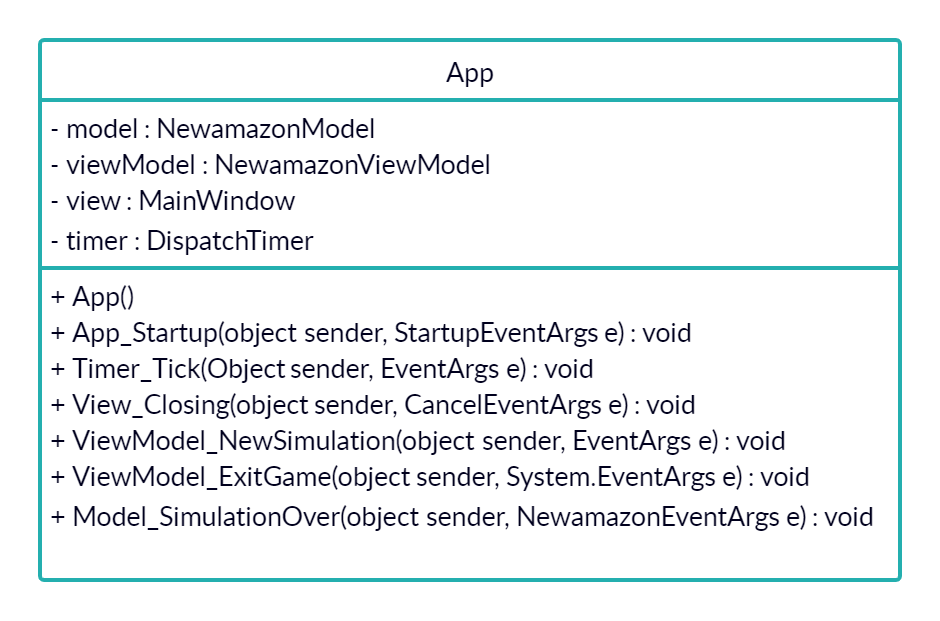
**Viewmodel:**

* DelegateCommand.cs
* NewamazonField.cs
* NewamazonViewModel.cs
* ViewModelBase.cs



**App:**

* App.xaml.cs



**NewamazonTest:**

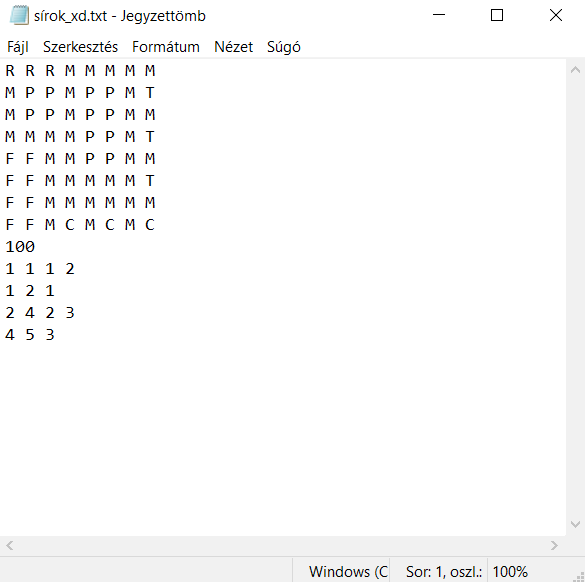
* NewamazonModelTest.cs

**Osztályok kifejtése**

**Model**:

* **NewamazonModel.cs:**
  + A központot tartalmazó osztály. A nézetmodellel eventek segítségével kommunikál. Számolja a lépések számát, nézi, hogy az egyes robotok mennyi energiát fogyasztott, illetve az összes energia fogyasztást.
* **NewamazonEventArgs.cs:**
  + Az eventek példányosítására létrehozott osztály, az EventArgs osztályból származik.
* **NewamazonClass.cs:**
  + A mezők osztályainak a base osztálya. Tartalmazza többek között az adott mező koordinátáit és IDját.
* **Mezo.cs**
  + A mezők példányosítására létrehozott osztály, a NewamazonClass osztályból származik. Tartalmazza, hogy éppen van-e rajta robot.
* **Polc.cs:**
  + A polcok példányosítására létrehozott osztály, a NewamazonClass osztályból származik. Tartalmazza, hogy az adott polcnak éppen melyik célállomáshoz kell menni, hogy van-e alatta robot és hogy hol a helye. Ha éppen nincs a helyén (egy robot viszi éppen), akkor mozog a robottal.
* **Robot.cs:**
  + A robotok példányosítására létrehozott osztály, a NewamazonClass osztályból származik. Tartalmazza az adott robot üzemanyagát, irányát és hogy van-e rajta szekrény.
* **Toltoallomas.cs:**
  + A töltőállomások példányosítására létrehozott osztály, a NewamazonClass osztályból származik. Tartalmazza, hogy foglalt-e vagy sem.
* **Celallomas.cs**
  + A célállomások példányosítására létrehozott osztály, a NewamazonClass osztályból származik. Tartalmazza, hogy foglalt-e vagy sem.
* **Kozpont.cs:**
  + A központ osztálya. Ez felel a robotok irányításáért, ő találja ki a robotok útját, illetve tárolja is azokat. A potenciális ütközéseket észreveszi, és igyekszik ilyenkor a lehető legkevesebb időveszteséggel megoldani azokat.
* **AllData.cs:**
  + A szimuláció fájlból való betöltéshez szükséges osztály, tartalmazza a szükséges kezdőadatokat (például a mezők kezdőpozícióit.)
* **NewamazonDataExcepion.cs:**
  + Az adatelérési kivétel példányosítására létrehozott osztály.
* **NewamazonFileDataAccess.cs:**
  + A szimuláció betöltéséért felelő osztály, az INewamazonDataAccess.cs osztályból származik.
* **INewamazonDataAccess.cs:**
  + Ebből az osztályból származik a NewamazonFileDataAccess.cs
* **MainWindow.xaml.cs:**
  + A szimuláció ablaka.
* **DelegateCommand.cs:**
  + Az általános parancs típusa.
* **NewamazonField.cs:**
  + Általános mező entitás típus. A játékmező számára biztosított osztály, amely eltárolja a pozíciót, szöveget, színt, irányt (robot esetén szükséges), stb. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe.
* **NewamazonViewModel.cs:**
  + A nézetmodell osztálya, a nézet tulajdonságait kezeli. Parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, valamint a kilépéshez. A parancsokhoz események vannak kötve, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását, de csupán információkat kér le tőle, illetve a játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.
* **ViewModelBase.cs:**
  + A nézetmodell base osztálya.
* **App.xaml.cs:**
  + 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. Továbbá az időzítők is itt futnak.
* **NewamazonModelTest.cs:**
  + Osztály a szimuláció helyességének tesztelésére.

**Mintabemenet:**

R: Robot

M: Üres mező

P: Polc

T: Töltőállomás

F: Fal

C: Célállomás

Ezután az első sorban a robotok kezdőüzemanyaga, majd ezután a megfelelő koordinátán lévő polcokon lévő áruk célállomása. (Pl: 2 4 2 3: 2. sor, 4. oszlop, 2es és 3as célállomás)