

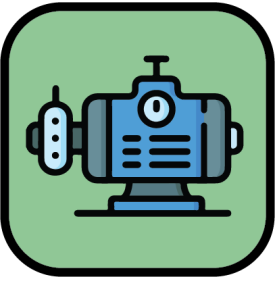
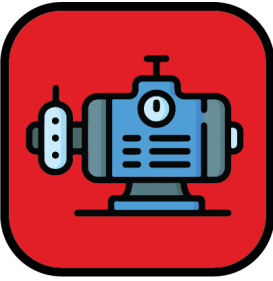
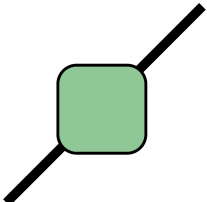
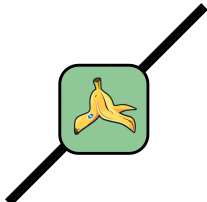
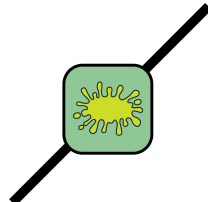
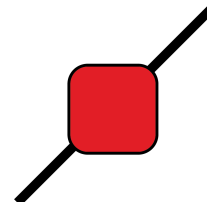
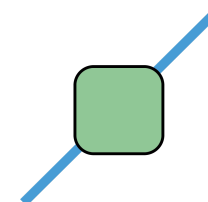


## 11. Grafikus felület specifikációja



### 11.1 A grafikus interfész

Pályaelemek:

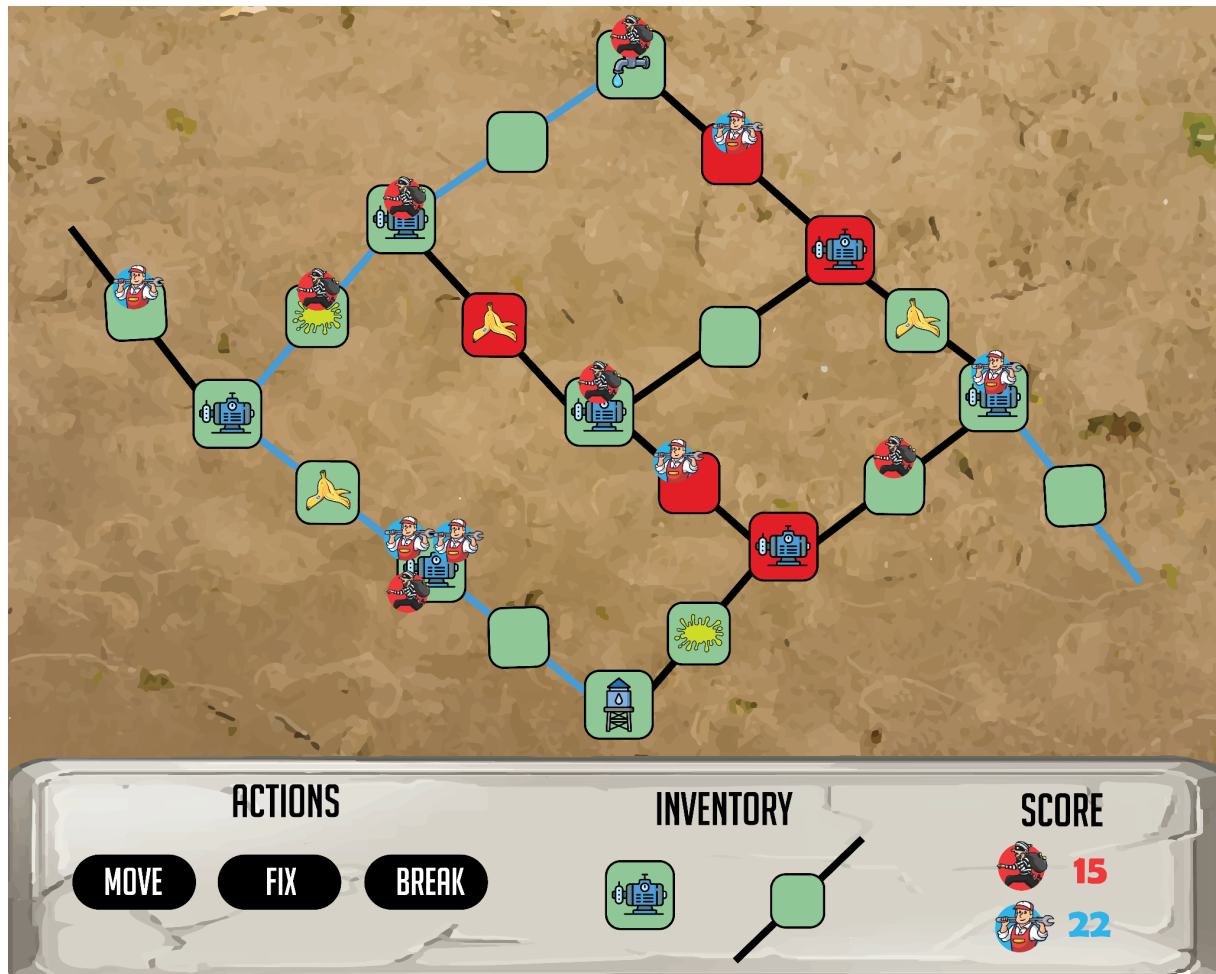
			
Ciszterna	Forrás	Pumpa	Törött pumpa

				
Cső	Csúszós cső	Ragadós cső	Lyukas cső	Vizes cső

Játékosok:

	
Szabotőr	Szerelő

### Látványterv:



## Interakciók a játék grafikus felületével

Ahogy a látványterven is látszik a játék kezelői felülete két fő részből áll: magából a hálózat reprezentálásából, illetve az alsó menüpanelből. Az utóbbi további két részre oszlik: a panel bal fele tartalmazza a kiválasztott játékos (szerelő vagy szabotőr) lehetséges akcióit, a másik fele az adott kiválasztott játékos eszköztárában található elemeket, amelyeket ha akar és megengedett, lehelyezhet.

A játékot az egér segítségével tudjuk vezérelni. Például egy játékos léptetését úgy tudjuk kivitelezni, hogy kiválasztjuk a játékost, majd kiválasztjuk a mezőt ahova el akarjuk léptetni, és a végén kiválasztjuk a megfelelő akciót, ami ebben az esetben a MOVE.

Azokhoz a műveletekhez, amelyeket csak az adott mezőn tud elvégezni a játékos, amelyen éppen áll elegendő kiválasztani a játékost, majd a lehetséges akciók közül kiválasztani egyet, például BREAK.

A csövek átállítást négy lépésben kell elvégezni. Például rákattintunk egy szerelőre, majd kiválasztjuk az új bemeneti csövet, ezután az új kimeneti csövet és végül kiválasztjuk a CHANGE FLOW akciót.

A játék állását a képernyő jobb sarkában lévő pontszám jelöli, amelyek az elfolyt illetve ciszternákba eljuttatott vizet jelképezik.

A játék nincs felosztva körökre, bárki bármikor végezhet akciót, akár egymás után többet is.

## 11.2 A grafikus rendszer architektúrája

### 11.2.1 A felület működési elve

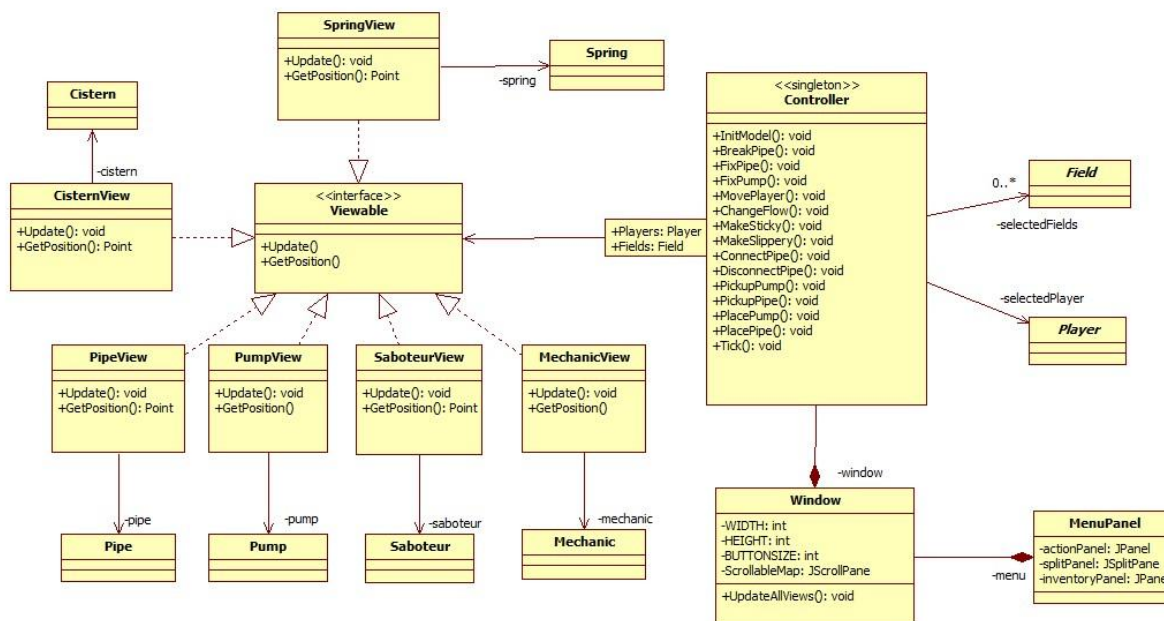
A megoldás során az **MVC** tervezési minta alapján dolgoztunk. A Controller osztályunk inicializálja a program indulása után a hálózatot, és ő teremti meg a kapcsolatot a különböző nézetek és a modell között.

A különböző nézeteknek egy tagváltozón keresztül van referenciájuk a modellbeli objektumra, amit grafikusán megjelenítenek, így tudják a modell objektum különböző állapotainak lekérdezésével azt megjeleníteni.

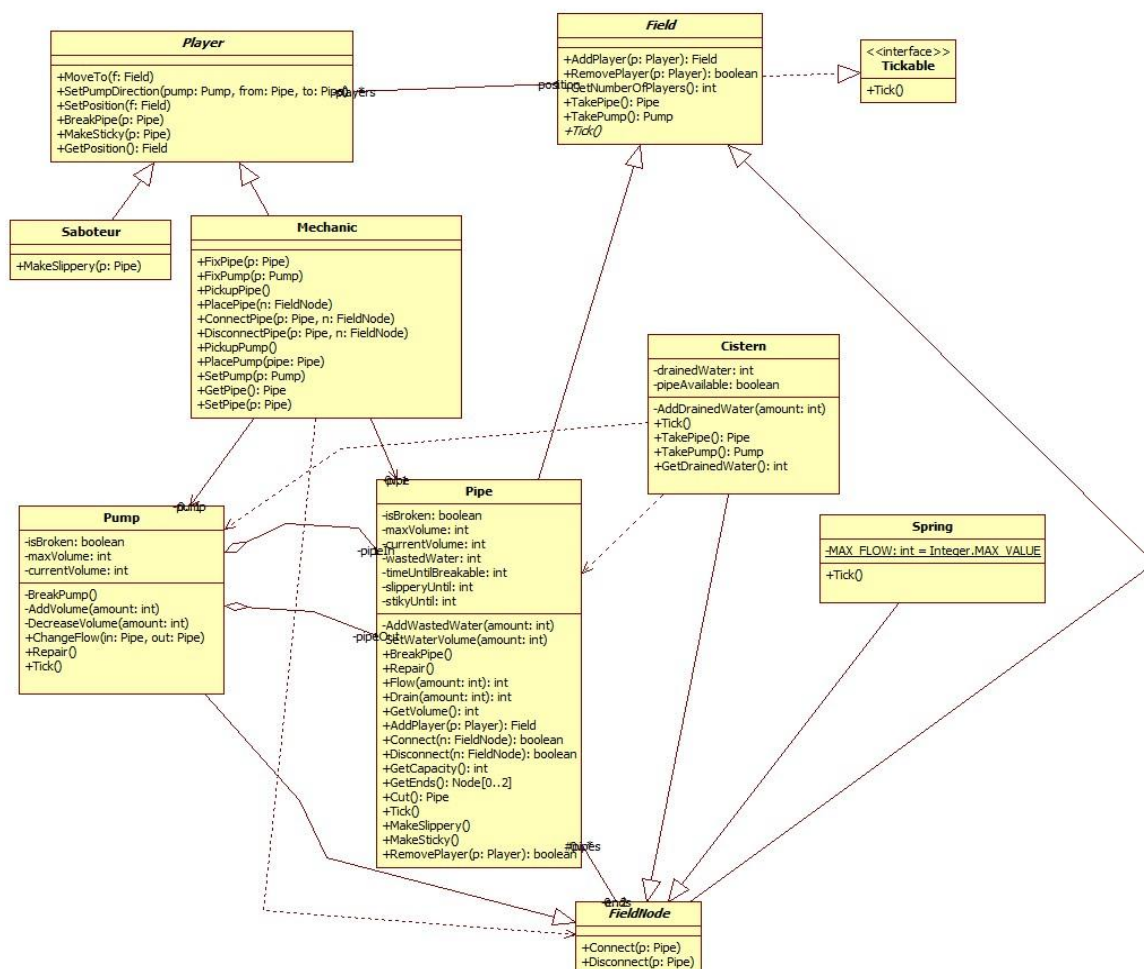
A különböző nézetek implementálják a *Viewable* interfészt ezáltal megvalósítják az *update()* metódust, melynek hatására újra kirajzolódnak. Minden nézetnek az ő osztálya a *JButton* osztály, és ezáltal fel tudunk venni nekik egy eseménykezelőt, amely a gomb megnyomásakor meghívja a Controller osztály megfelelő függvényét ami beregisztrálja, hogy az adott gomb meg lett nyomva. Az menüpanelben lévő akciók megnyomására is meghívódik a *Controller* megfelelő függvénye, és mivel már be voltak regisztrálva az előzőekben megnyomott gombok, így azoknak függvényében meg tudja hívni a modell megfelelő metódusait, majd meghívja a *window* tagváltozója *updateAllViews()* metódusát és így az összes nézet lekéri a modelljétől az új állapotát és ennek függvényében rajzolja ki magát.

Így tehát mi alapvetően a **pull** modellt alkalmazzuk, mert a nézetek mindig lekérdezik a modell állapotát és annak függvényében rajzolják ki a megfelelő megjelenítést. Ehhez a modellen annyit kellett változtatni, hogy ahol nem voltak **getter** függvények, ott kiegészítettük az osztályokat ezekkel.

## 11.2.2 A felület osztály-struktúrája



## 11.2.3 A módosított modell osztálydiagramja



## 11.3 A grafikus objektumok felsorolása

### 11.3.1 Controller

- **Felelősség**

A Controller osztály felelős a felhasználói események kezeléséért, ezek továbbításáért a modell felé, a modell létrehozásáért és view létrehozásáért.

- **Össztályok**

-

- **Interfészek**

-

- **Attribútumok**

- **+HashMap<Player, Viewable> players:** a hálózatban található játékosok modellbeli objektumait mappeli a hozzájuk tartozó nézethez
- **+HashMap<Field, Viewable> fields:** a hálózatban található mezők modellbeli objektumait mappeli a hozzájuk tartozó nézethez
- **-Player selectedPlayer:** a kiválasztott játékos, aki az akciót végezni fogja
- **-Field[] selectedFields:** a kiválasztott mezők, amelyek az akció részei
- **-Window window:** az ablak ahol megjelenítjük a hálózatot és a menüket

- **Metódusok**

- **+void movePlayer():** Az aktuálisan kiválasztott játékost rálépteti a kiválasztott mezőre.
- **+void changeFlow():** Az aktuálisan kiválasztott szerelőn meghívja a setPumpDirection() függvényt így átállítva a pumpa ki- és bemenetét. Az első kiválasztott csövet adja meg mint bemenet és a másodikat mint kimenet.
- **+void fixPump():** Az aktuálisan kiválasztott szerelőn meghívja a fixPump() függvényt ezzel megjavítva a pumpát amin a szerelő áll.
- **+void breakPipe():** Az aktuálisan kiválasztott játékoson meghívja a breakPipe() függvényt ezzel eltörve a csövet amin éppen áll.
- **+void fixPipe():** Az aktuálisan kiválasztott szerelőn meghívja a fixPipe() függvényt ezzel megjavítva a csövet amin éppen áll.
- **+void makeSticky():** Az aktuálisan kiválasztott játékoson meghívja a makeSticky() függvényt ezzel ragadóssá téve a csövet amin éppen áll.
- **+void makeSlippery():** Az aktuálisan kiválasztott szabotőrön meghívja a makeSlippery() függvényt ezzel csúszóssá téve a csövet amin éppen áll.
- **+void connectPipe():** Az aktuálisan kiválasztott szerelő felcsatlakoztatja a megadott szabad végű csövet a kiválasztott csomópontához, a szerelő connectPipe() függvénye segítségével.
- **+void disconnectPipe():** Az aktuálisan kiválasztott szerelő lecsatlakoztatja a megadott csövet a kiválasztott csomóponttól, a szerelő disconnectPipe() függvénye segítségével.
- **+void pickupPipe():** Az aktuálisan kiválasztott szerelőn meghívja a pickupPipe() függvényt ezzel felvéve egy csövet a ciszternától amin éppen áll.
- **+void pickupPump():** Az aktuálisan kiválasztott szerelőn meghívja a pickupPump() függvényt ezzel felvéve egy pumpát a ciszternától amin éppen áll.

- **+void placePump()**: Az aktuálisan kiválasztott szerelő leteszi a nála lévő pumpát a csőre amin éppen áll, a szerelő placePump() függvénye segítségével.
- **+void placePipe()**: Az aktuálisan kiválasztott szerelő leteszi a nála lévő csövet a kiválasztott és azon csomópont közé amin éppen áll, a szerelő placePipe() függvénye segítségével.
- **+void initModel()**: Felépít egy kezdetleges pályát, elkészíti a pályaelemeket és a hozzájuk tartozó View elemeket majd ezeket egymáshoz köti.
- **+void Tick()**: Meghívja az összes játékelem Tick() metódusát és modellezi a víz folyását.

### 11.3.2 MenuPanel

- **Felelősség**

A MenuPanel az eszköztár, illetve a különböző elvégezhető akciók megjelenítéséért felelős. Létrehozza a különböző gombokat, inicializálja őket és felveszi hozzájuk a megfelelő eseménykezelőket, hogy azok kommunikálhassanak megfelelően majd a *Controller*-el.

- **Ősosztályok**

JPanel.

- **Interfészek**

-

- **Attribútumok**

- - **JPanel actionPanel**: panel, amely a különböző akciók gombjait tartalmazza
- - **JPanel inventoryPanel**: panel, amely egy játékos eszköztárában található elemeket tartalmazza
- - **JSplitPane splitPanel**: split panel, amely az *actionPanel*-t és az *inventoryPanel*-t tartalmazza

- **Metódusok**

-

### 11.3.3 Window

- **Felelősség**

A program ablakának megjelenítéséért, illetve a különböző nézetek frissítésért felelős osztály.

- **Ősosztályok**

JFrame.

- **Interfészek**

-

- **Attribútumok**

- - **int WIDTH**: a program ablakának szélessége.
- - **int HEIGHT**: a program ablakának magassága.

- **- *int* *BUTTONSIZE***: A gombok mérete.
- **- *JScrollPane scrollableMap***: görgethető panel, amelyen a hálózatot jelenítjük meg
- **- *MenuPanel menu***: a menüpanel amely tartalmazza az eszköztárat és akciókat
- **Metódusok**
  - **+void *updateAllViews()***: a *Controller* osztályban tárolt összes nézetnek meghívja az *update()* függvényét, és frissíti az összes nézetet.

#### 11.3.4 Viewable

- **Felelősség**

Interfész amelyet azok a modellbeli osztályok valósítanak meg amelyeknek van nézetük, és valamiképpen megjelennek grafikusán a hálózatban.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**

-

- **Metódusok**

- **+ void *update()***: a nézet frissítése a modellbeli objektum állapota szerint
- **+ *Point* *getPosition()***: visszaadja a nézet koordinátáit

#### 11.3.5 CisternView

- **Felelősség**

Egy ciszterna grafikus reprezentációja. Felelős a *Controller* megfelelő metódusának meghívásáért, amikor click esemény történik az adott ciszternán. Illetve felelős a ciszterna megfelelő kirajzolásáért a modellbeli ciszterna állapotától függően.

- **Ősosztályok**

*JButton*.

- **Interfészek**

*Viewable*.

- **Attribútumok**

- **-*Cistern cistern***: a modellbeli *Cistern* objektum, amelyet megjelenít a *CisternView*

- **Metódusok**

- **+void *update()***: a nézet frissítése a modellbeli objektum állapota szerint
- **+*Point* *getPosition()***: visszaadja a nézet koordinátáit

### 11.3.6 PipeView

- **Felelősség**

Egy cső grafikus reprezentációja. Felelős a Controller megfelelő metódusának meghívásáért, amikor click esemény történik az adott csőn. Illetve felelős a cső megfelelő kirajzolásáért a modellbeli cső állapotától függően.

- **Ősosztályok**

JButton.

- **Interfészek**

Viewable.

- **Attribútumok**

- - **Pipe pipe**: a modellbeli *Pipe* objektum, amelyet megjelenít a *PipeView*

- **Metódusok**

- +**void update()**: a nézet frissítése a modellbeli objektum állapota szerint
- +**Point getPosition()**: visszaadja a nézet koordinátáit

### 11.3.7 PumpView

- **Felelősség**

Egy pumpa grafikus reprezentációja. Felelős a Controller megfelelő metódusának meghívásáért, amikor click esemény történik az adott pumpán. Illetve felelős a pumpa megfelelő kirajzolásáért a modellbeli pumpa állapotától függően.

- **Ősosztályok**

JButton.

- **Interfészek**

Viewable.

- **Attribútumok**

- - **Pump pump**: a modellbeli *Pump* objektum, amelyet megjelenít a *PumpView*

- **Metódusok**

- +**void update()**: a nézet frissítése a modellbeli objektum állapota szerint
- +**Point getPosition()**: visszaadja a nézet koordinátáit

### 11.3.8 SpringView

- **Felelősség**

Egy forrás grafikus reprezentációja. Felelős a Controller megfelelő metódusának meghívásáért, amikor click esemény történik az adott forráson. Illetve felelős a forrás megfelelő kirajzolásáért a modellbeli forrás állapotától függően.

- **Ősosztályok**

JButton.



- **Interfészek**

Viewable.

- **Attribútumok**

- - **Spring spring**: a modellbeli *Spring* objektum, amelyet megjelenít a *SpringView*

- **Metódusok**

- +**void update()**: a nézet frissítése a modellbeli objektum állapota szerint
- +**Point getPosition()**: visszaadja a nézet koordinátáit

### 11.3.9 MechanicView

- **Felelősség**

Egy szerelő grafikus reprezentációja. Felelős a Controller megfelelő metódusának meghívásáért, amikor click esemény történik az adott szerelőn. Illetve felelős a szerelő megfelelő kirajzolásáért a megfelelő mezőre, a modellbeli szerelő állapotától függően.

- **Össztályok**

JButton.

- **Interfészek**

Viewable.

- **Attribútumok**

- -**Mechanic mechanic**: a modellbeli *Mechanic* objektum, amelyet megjelenít a *MechanicView*

- **Metódusok**

- +**void update()**: a nézet frissítése a modellbeli objektum állapota szerint
- +**Point getPosition()**: visszaadja a nézet koordinátáit

### 11.3.10 SaboteurView

- **Felelősség**

Egy szabotőr grafikus reprezentációja. Felelős a Controller megfelelő metódusának meghívásáért, amikor click esemény történik az adott szabotőrön. Illetve felelős a szabotőr megfelelő kirajzolásáért a megfelelő mezőre, a modellbeli szabotőr állapotától függően.

- **Össztályok**

JButton.

- **Interfészek**

Viewable.

- **Attribútumok**

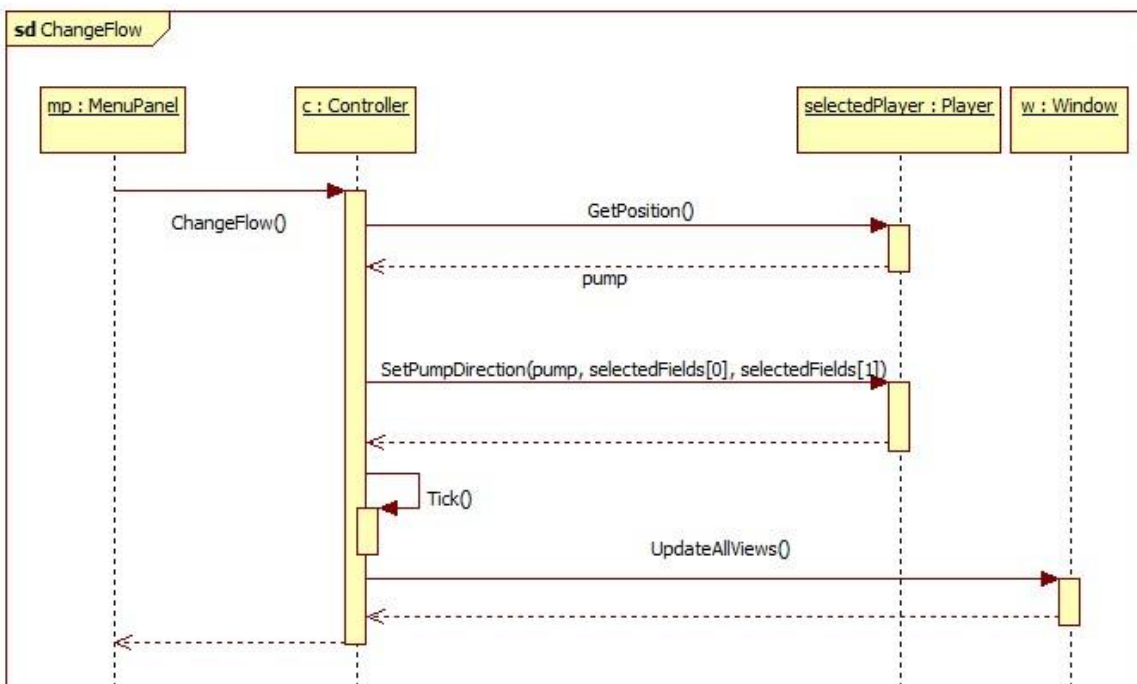
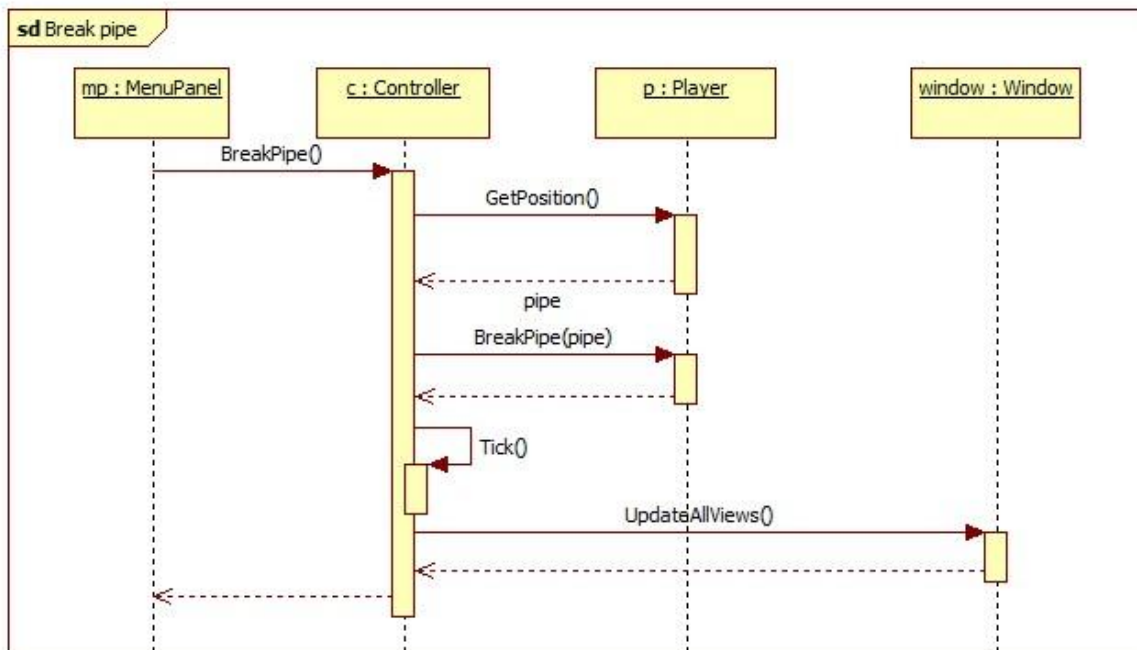
- -**Saboteur saboteur**: a modellbeli *Saboteur* objektum, amelyet megjelenít a *SaboteurView*

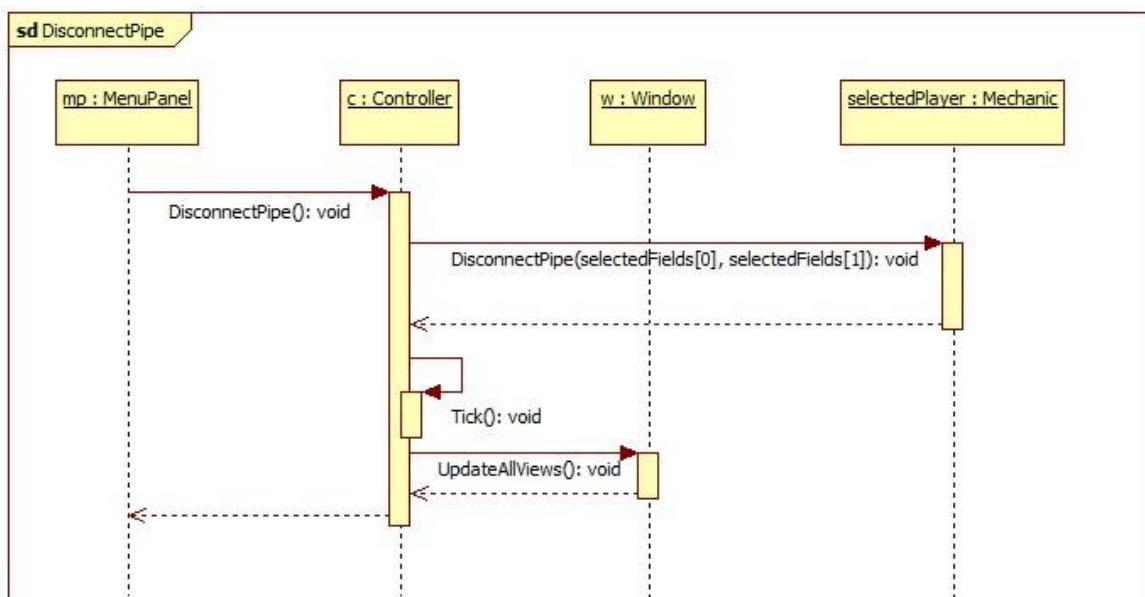
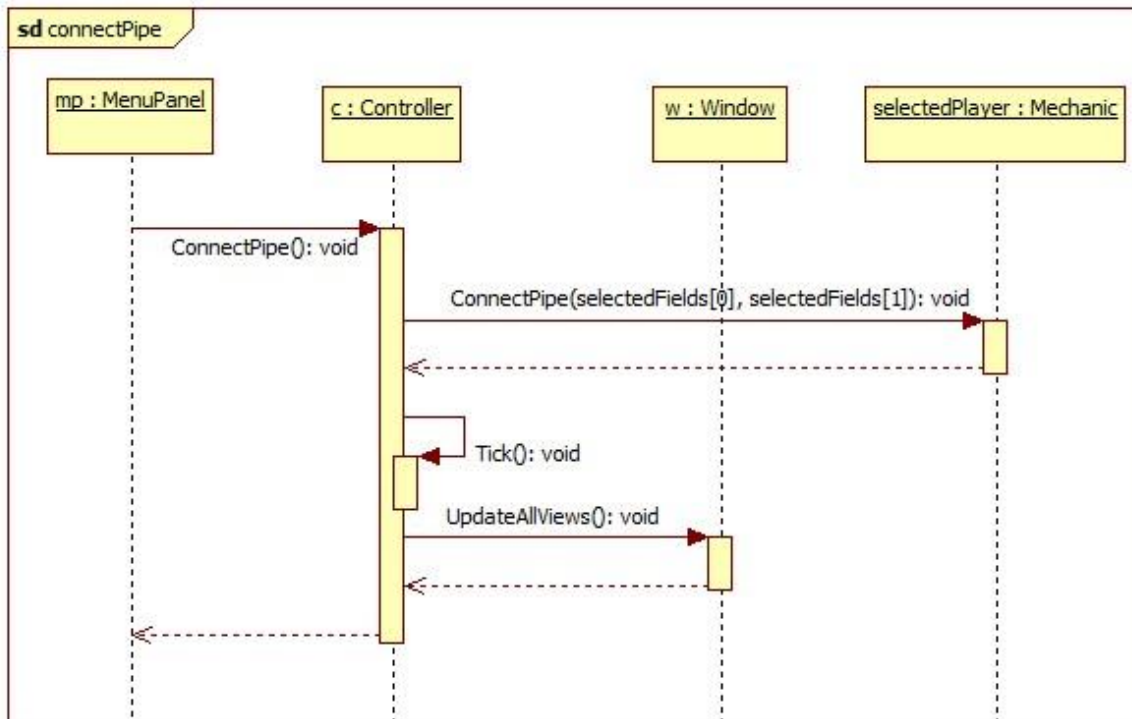
- **Metódusok**

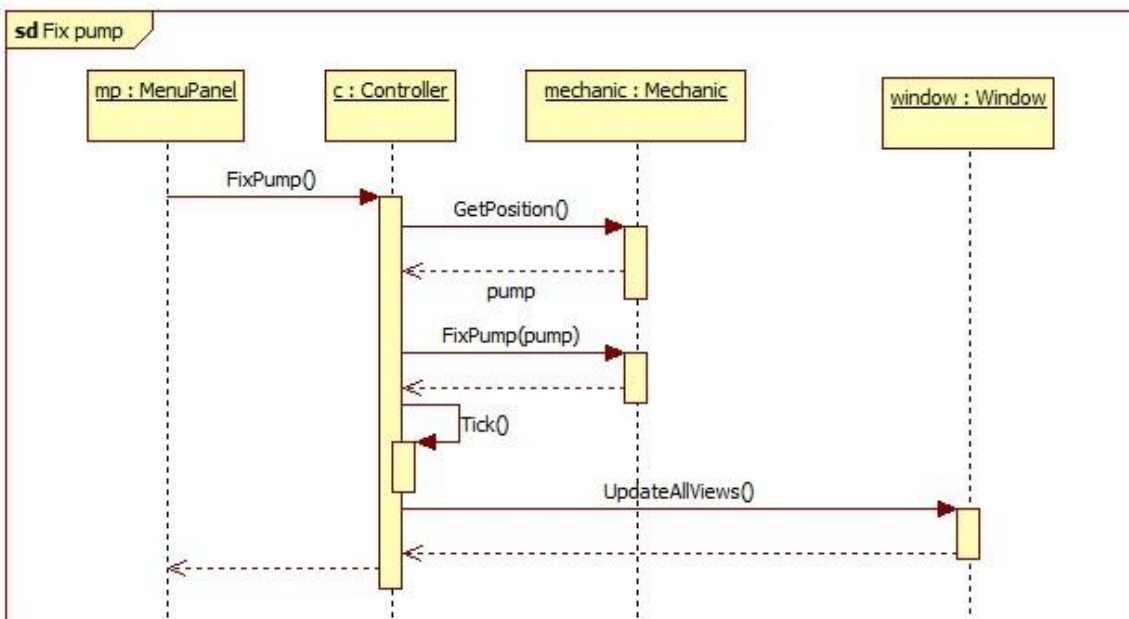
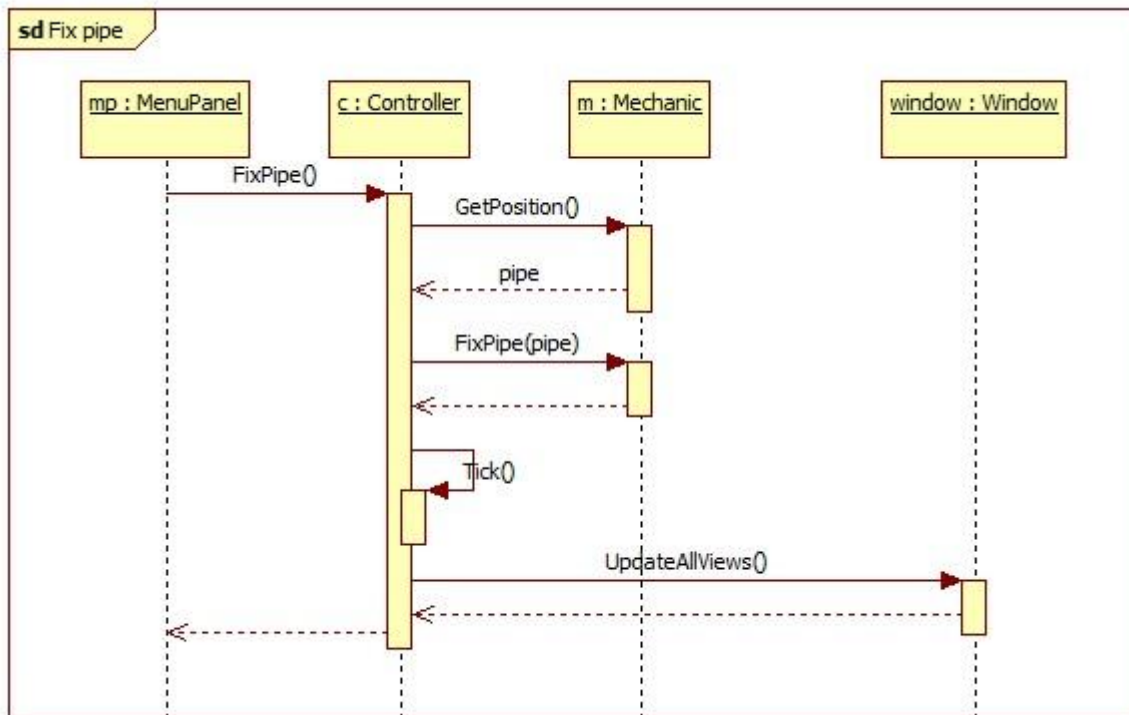
- **+Point getPosition():** a nézet frissítése a modellbeli objektum állapota szerint
- **+void update():** visszaadja a nézet koordinátáit

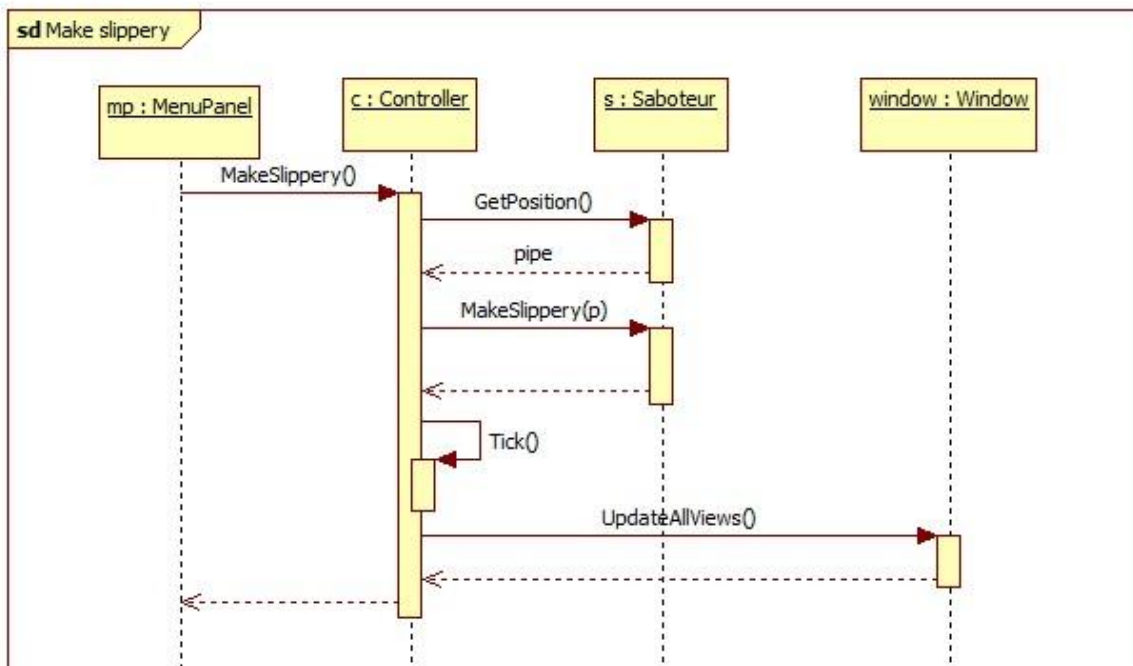
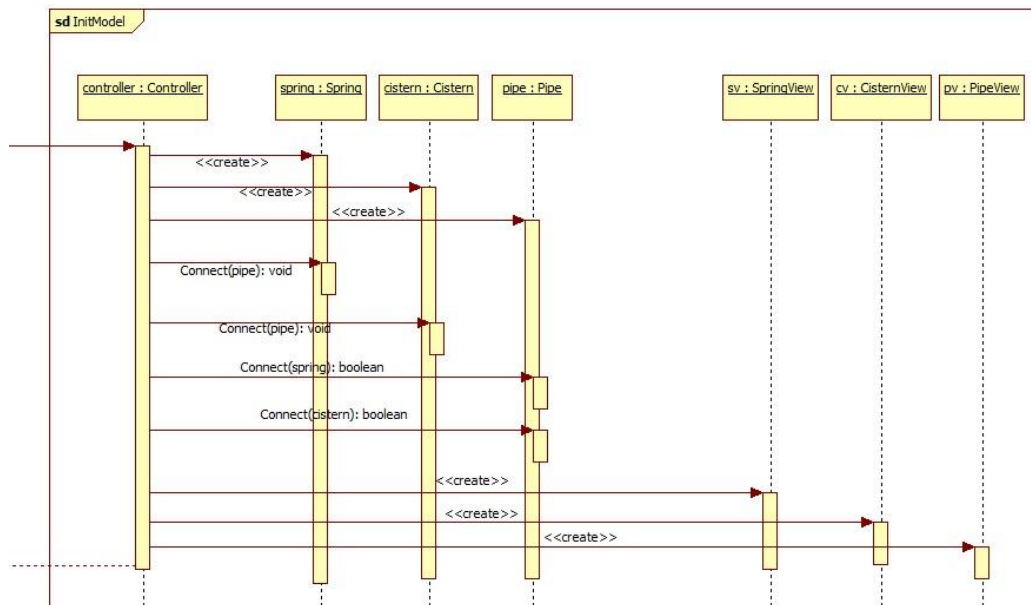
## 11.4 Kapcsolat az alkalmazói rendszerrel

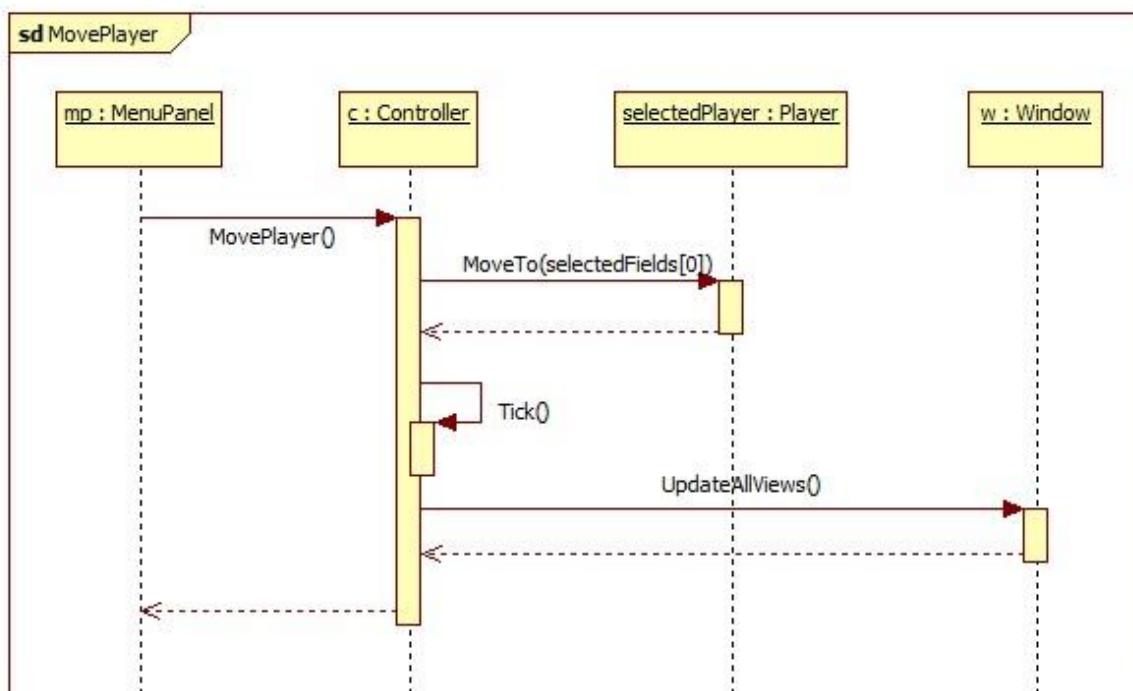
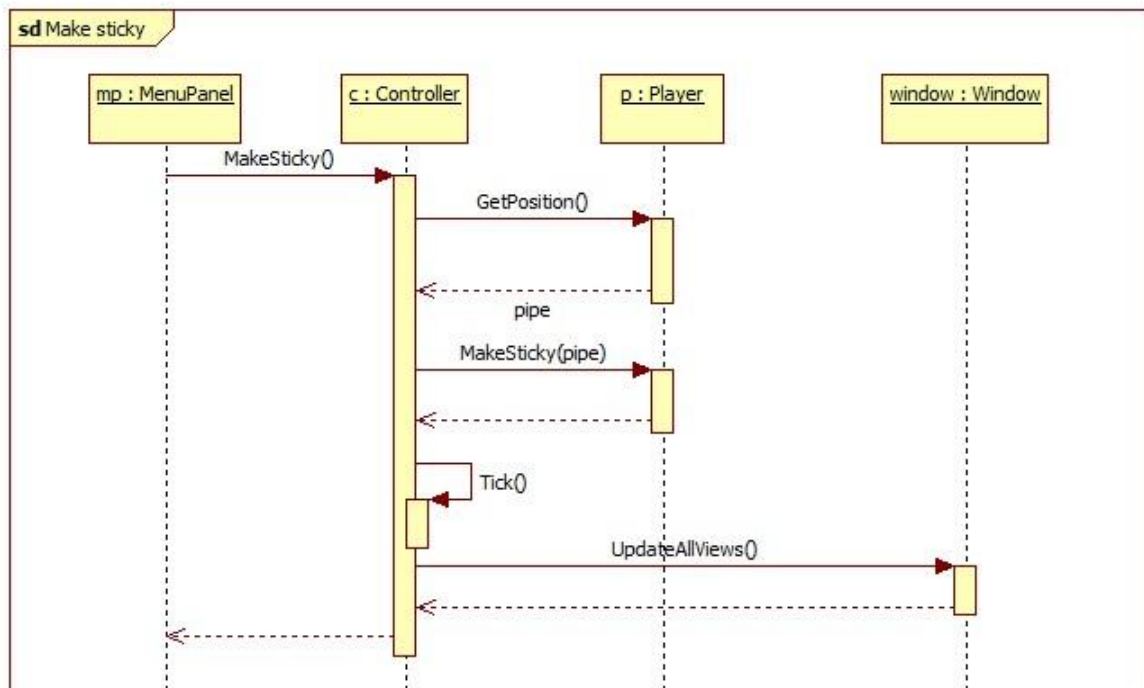
### 11.4.1 Szekvencia diagramok

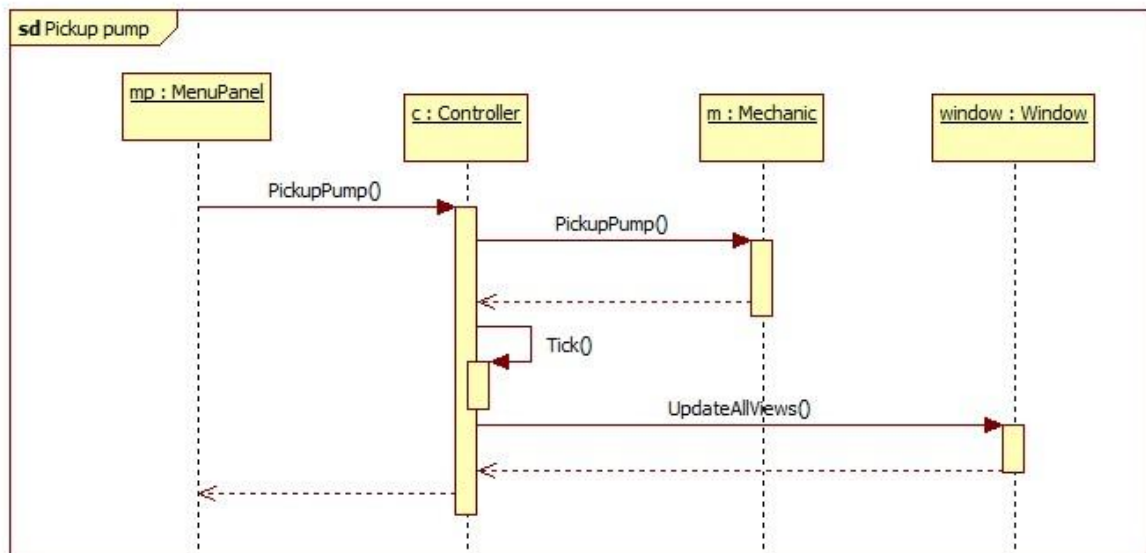
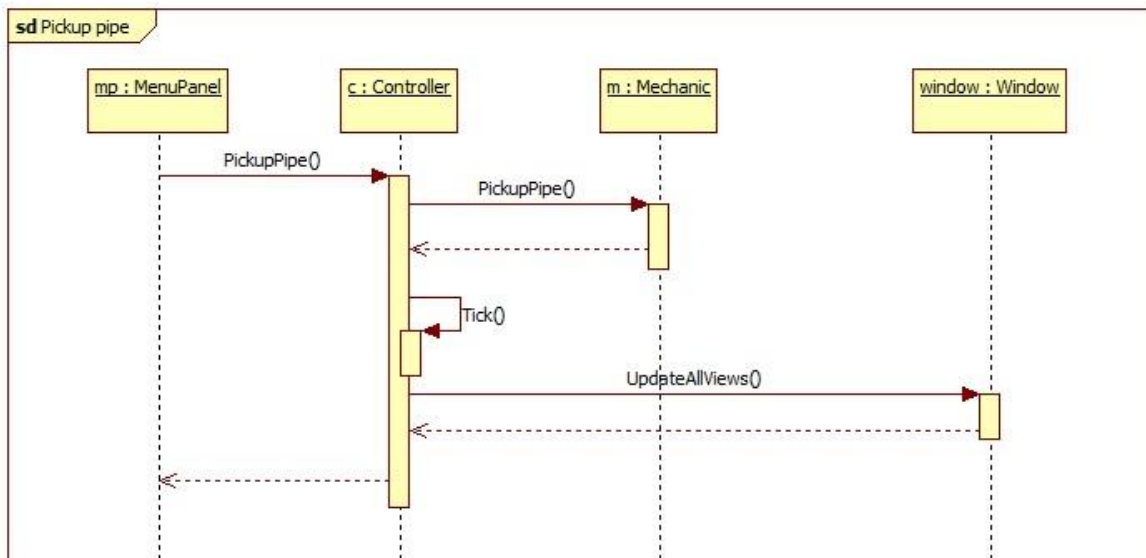


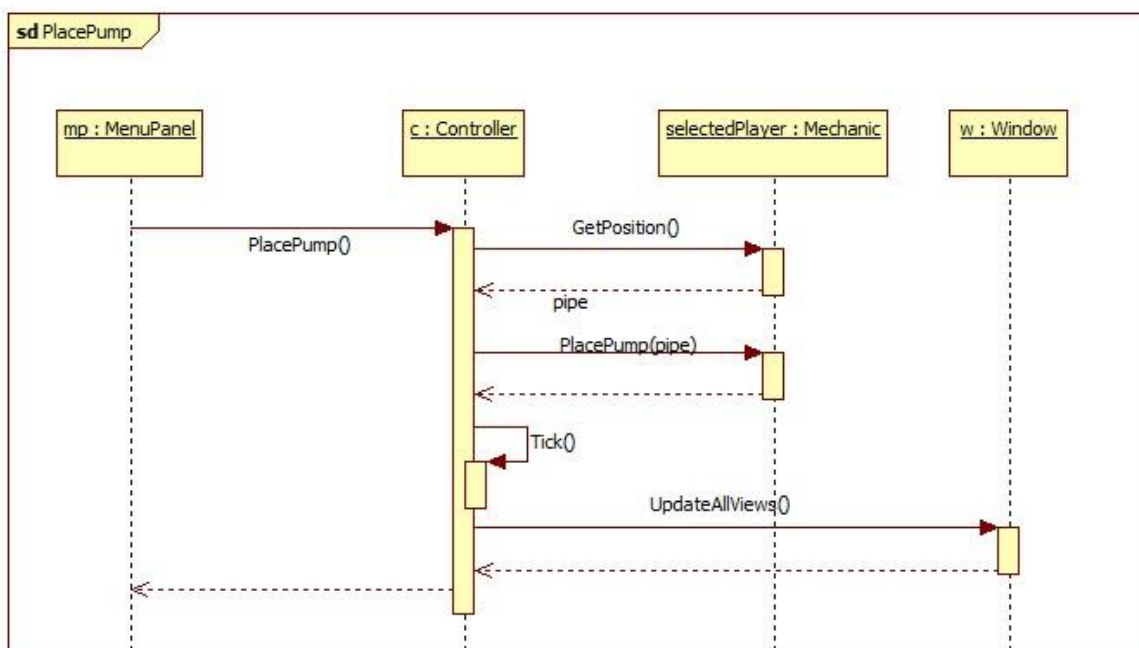
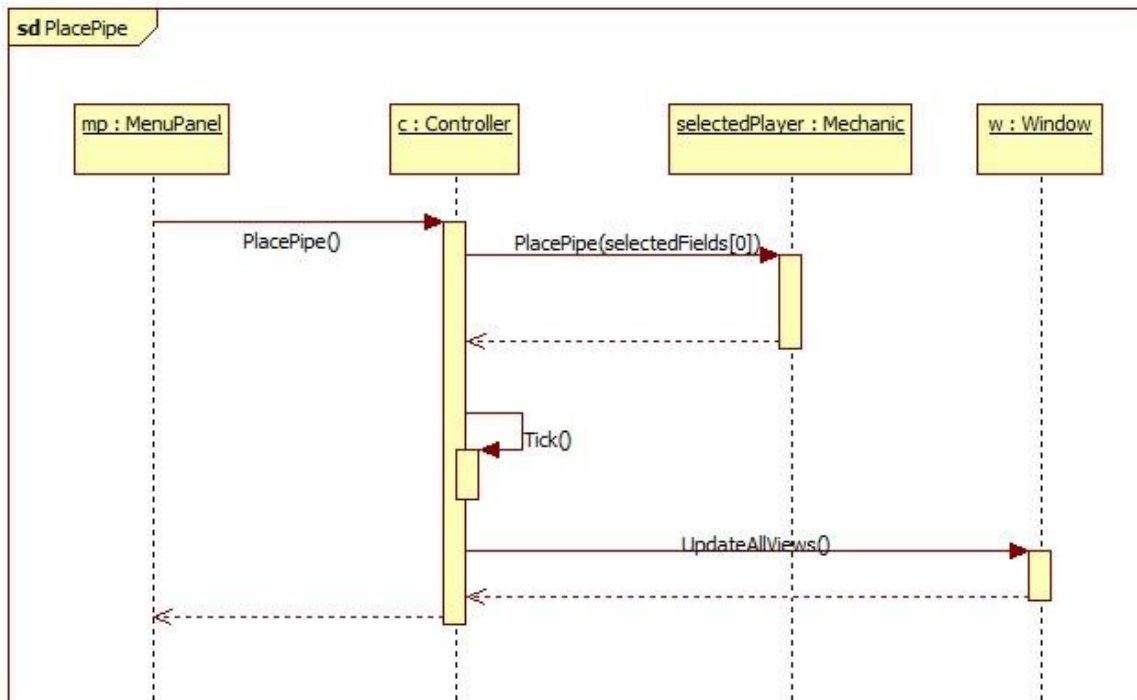














**11.5 Napló**

<b>Kezdet</b>	<b>Időtartam</b>	<b>Résztevők</b>	<b>Leírás</b>
2023.05.17. 18:00	2,5 óra	Palásti Nagy Kurcsi Barabási Ganzer	<b>Értekezlet.</b> <b>Döntés:</b> Grafikus felület megjelenésének megtervezése.
2023.05.18. 10:00	3 óra	Nagy	<b>Tevékenység:</b> Ikonok készítése.
2023.05.19. 18:00	1 óra	Palásti Nagy Kurcsi Barabási Ganzer	<b>Értekezlet.</b> <b>Döntés:</b> Grafikus felület architektúrája, osztálydiagram megtervezése.
2023.05.20. 10:00	1 óra	Nagy	<b>Tevékenység:</b> Látványterv készítése
2023.05.20. 11:00	30 perc	Kurcsi	<b>Tevékenység:</b> Látványterv leírás
2023.05.20. 15:00	1 óra	Kurcsi	<b>Tevékenység:</b> Osztálydiagram elkészítése
2023.05.20. 19:00	3 óra	Palásti	<b>Tevékenység:</b> Szekvencia diagramok készítése
2023.05.20. 19:00	1 óra	Barabási	<b>Tevékenység:</b> Dokumentum szerkesztése (11.3.1, 11.3.2, 11.3.3 )
2023.05.20. 20:00	2 óra	Kurcsi	<b>Tevékenység:</b> Dokumentum szerkesztése: (11.3.4, 11.3.5, 11.3.6, 11.3.7, 11.3.8, 11.3.9, 11.3.10 )
2023.05.20. 21:00	1 óra	Palásti	<b>Tevékenység:</b> Osztálydiagram javítása, változtatások átvezetése szekvencia diagramokban.

2023.05.21. 10:00	3 óra	Barabási	<b>Tevékenység:</b> Szekvencia diagramok készítése
2023.05.21. 11:00	30 perc	Kurcsi	<b>Tevékenység:</b> 11.2.1
2023.05.21.13:00	2 óra	Ganzer	<b>Tevékenység:</b> Szekvencia diagramok javítása
2023.05.21. 17:00	1,5 óra	Ganzer	<b>Tevékenység:</b> Dokumentum ellenőrzése, hibák javítása, véglegesítés