# Házi feladat specifikáció

Balassa Ádám DXXEXO | A programozás alapjai III. | Pandemic

## A feladat leírása

A Pandemic egy 2-4 fős társasjáték, a házifeladatom ennek a JAVA nyelven való implementációja. A játék célja, hogy a játékosok a „tábla" ellen összefogva próbáljanak elfojtani 4 járványt.

A játék szabályai elég összetettek, így bár itt összefoglalom a lényegét, részletes leírást ad az alábbi PDF.
https://images-cdn.zmangames.com/us-east-1/filer_public/25/12/251252dd-1338-4f78-b90d-afe073c72363/zm7101_pandemic_rules.pdf

A játékosok egy a világ metropoliszaiból álló gráfon mozoghatnak. Mindegyiküknek van legfeljebb 7db kártyája, melyek a városokra mutatnak, mozogni többek közt ezek kijátszásával képesek. A 4 féle járvány 4 szín által van reprezentálva, minden városnak is van egy ilyen színe. Minden játékos 4 akciót tehet a körében, mely lehet mozgás (szomszéd mezőre, kutató állomásról kutató állomásra, kártyát tartalmazó mezőre, vagy bármely mezőre ha a játékos azt a kártyát dobja el, amin áll). Ha eldobja azt a kártyát amilyen mezőn áll, képes még kutatóállomás építésére, mely a játékban „teleportkapuként" funkcionál. A játékos képes ezen kívül 1db fertőzés irtására azon a mezőn amin áll. Végül, és ami a játék céja is, képes egy kutatóállomáson állva 5 ugyanolyan színű kártya eldobásával az adott színű ellenszer kifejlesztésére. Ha egy színből ki van fejlszetve az ellenszer, a játékos 1 akcióval képes leszedni az ilyen színű összes fertőzést a mezőről amin áll.

A játékosok akkor nyertek, ha a 4 ellenszert kifejlesztették.

Minden játékos a köre végén 2 kártyát húz, illetve egy másik ugyanígy városokat tartalmazó pakliból (fertőzés pakli) is húz kettőt, ezen két város 1-1 vírussal megfertőződik. A játék kártyák közt, a városokon kívül, akció nélkül bármelyik pillanatban kijátszható segítő esemény kártyák is vannak, illetve 4/5/6db EPIDEMIC kártya.

Epidemic kártya húzásakor megnövekszik a fertőzöttségi szint (kezdetben 1-es, ha eléri a 3-mast, akkor minden kör végén 3 új város fertőződik, ha eléri az 5-öt, akkor 4). Ugyanekkor egy város megfertőződik egyszerre 3 vírussal és az eddig megfertőződött városok kártyái (vagyis a fertőzés kártyák dobó paklija) visszakeveredik a fertőzés pakli tetejére.

Ha egy város már 3 ugyanolyan színű vírussal fertőzött és újra fertőződne egy 4-edikkel, akkor ahelyett, hogy ez megtörténne, minden szomszédos város fertőződik meg 1-1-gyel. Ez egy láncreakciót indíthat kitörésekből, amelyekből, ha 8db megtörténik egy játékban, a játékosok vesztettek. Hasonlóan vesztettek, hogy ha egy színből kikerül túl sok (24db) vírus a pályára avagy, ha elfogy a húzópakli (kifutnak az időből).

A Java applikációm ezeket a szabályokat ismeri, és teljes mértékben automatizálja, a játékos feladata pusztán az akcióinak kiválasztása. Ezen kívül adott még néhány szabály (mik is vannak az esemény kártyákon, illetve minden játékos kap egy karakter kártyát melyeknek speciális képességei vannak, ezt feleslegesnek tartom ilyen részletességgel kifejteni), a játékom azokra is odafigyel.

## A játék megvalósítása

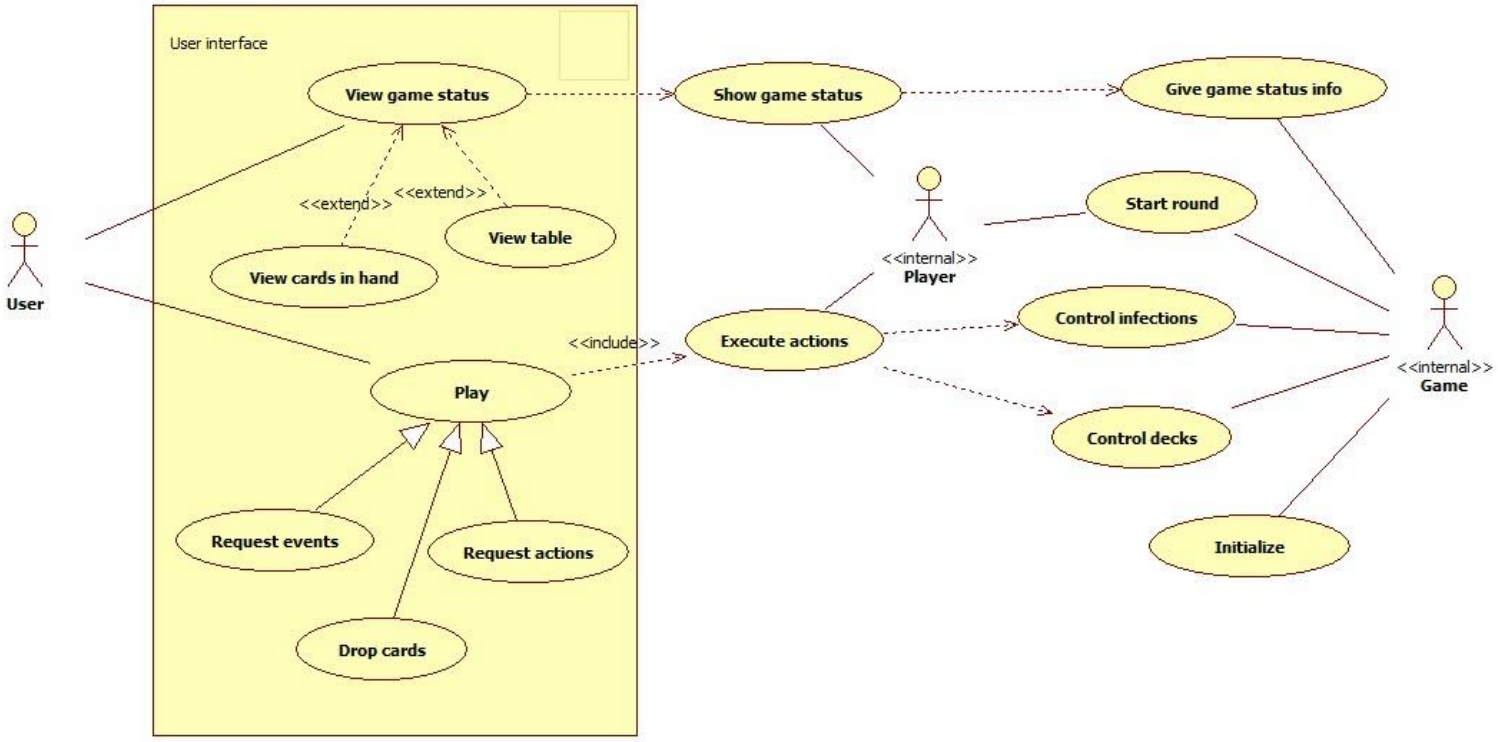
Egy garfikus felületen a játékosok beállítják a játék alap adatait (hány játékos játszik, milyen a játék nehézsége, vagyis hány Epidemic kártya keverődjön a pakliba). Ezután kiválaszthatják, hogy új játékot

akarnak indítani, vagy a korábban elmentettet kívánják folytatni (start / load). Ezután két lehetőség közül választhatnak: Console Application és Graphics Application. A kettő a felhasználó felé nyújtott interfészében különbözik egymástól, a játék belső logikája ugyanaz. A Console Applicationt biztosan megvalósítom, ott CMD-hez hasonló parancsokkal lehet irányítani a játékot és a standard input-output-on lehet vele kommunikálni. A Graphics Application lehetőség egy az eredetihez hasonló képet jelenítene meg a játékról és azt kattintásokkal lehetne irányítani.

Console Application-ként elkezdett (start) játékot el lehet indítani Graphics Applicationként (load) is.

A felhasználó felé mutatott interfészétől függetlenül a játék belsejében rögzítve vannak a szabályok, amely belső állapotát minden kör elején fájlba menti. A játékos a köre befejezése előtt visszavonhatja az abban körben tett lépéseit (újrakezdheti a körét). Ugyanebből a fájlból olvasva képes a játék visszaállítani egy korábban elkezdett verziót.

## Use case-ek




### Initialize

A játék képes magát felépíteni, létrehozni a szükséges objektumokat és rögzíteni a játékosokat

### Control decks

A játék felelőssége a fertőzés pakli, játék pakli és ezek dobó paklijainak kezelése. Képes húzni kártyát belőlük, odaadni azt egy Player-nek és Player által adott kártyát a dobó pakliba helyezni. Ezen use-case körbe tartozik a játék befejezése, ha elfogy a pakli.

### Control infections

A játék képes a saját tábláját fertőzni, a fertőzöttségi szintet és a kitöréseket kezelni. Ezen szerepkörbe tartozik a játék befejezése, ha túl sok fertőzés kerül a táblára, vagy túl sok kitörés történik

### Give game status info
A játék képes a Player kérésére általa ismert adatok megosztására

### Start round
A játék képes egy Player körének elindítására amely asszociációban van azzal, hogy egy Player képes erre a kérésre felépíteni a saját körét. A Player felelőssége számon tartani a hátralévő akcióinak számát

### Show game status
Egy Player egy User kérésére képes a játék adatainak lekérésére és továbbítására

### Execute actions
Egy Player képes egy user bármilyen vezérő interakcióját végrehajtani. Képes magát a táblán mozgatni, képes a kezében lévő kártyák kezelésére, vírusok irtására stb. Ezen képességének érvényesítéséhez a Game Control infections és Control Decks képességét is használja.

### Play
Egy felhasználó képes vezérlő utasításokra, melyet egy Player végrehajt

### Request events
Képes lehet eseményeket jelezni

### Request actions
Képes akciók végrehajtását kérni

### Drop cards
Kérheti a Player-ét, hogy játszon ki, dobjon el kártyákat

### View game status
Egy felhasználó képes megtekinteni a táblát, a Playerének kártyáit és egyéb infót amire szükség van (hátralevő akciók száma, fertőzöttségi szint stb.)

## Megjegyzés
Use case-ek szempontjából lényegtelen, hogy az internal controller is több részre oszlik, egy Core játékra és egy, a user-rel kommunikáló Player-re. Ez a modell egy finomítása annak, amit a User érzékel, már információt tartalmaz a játék belső logikájáról is.