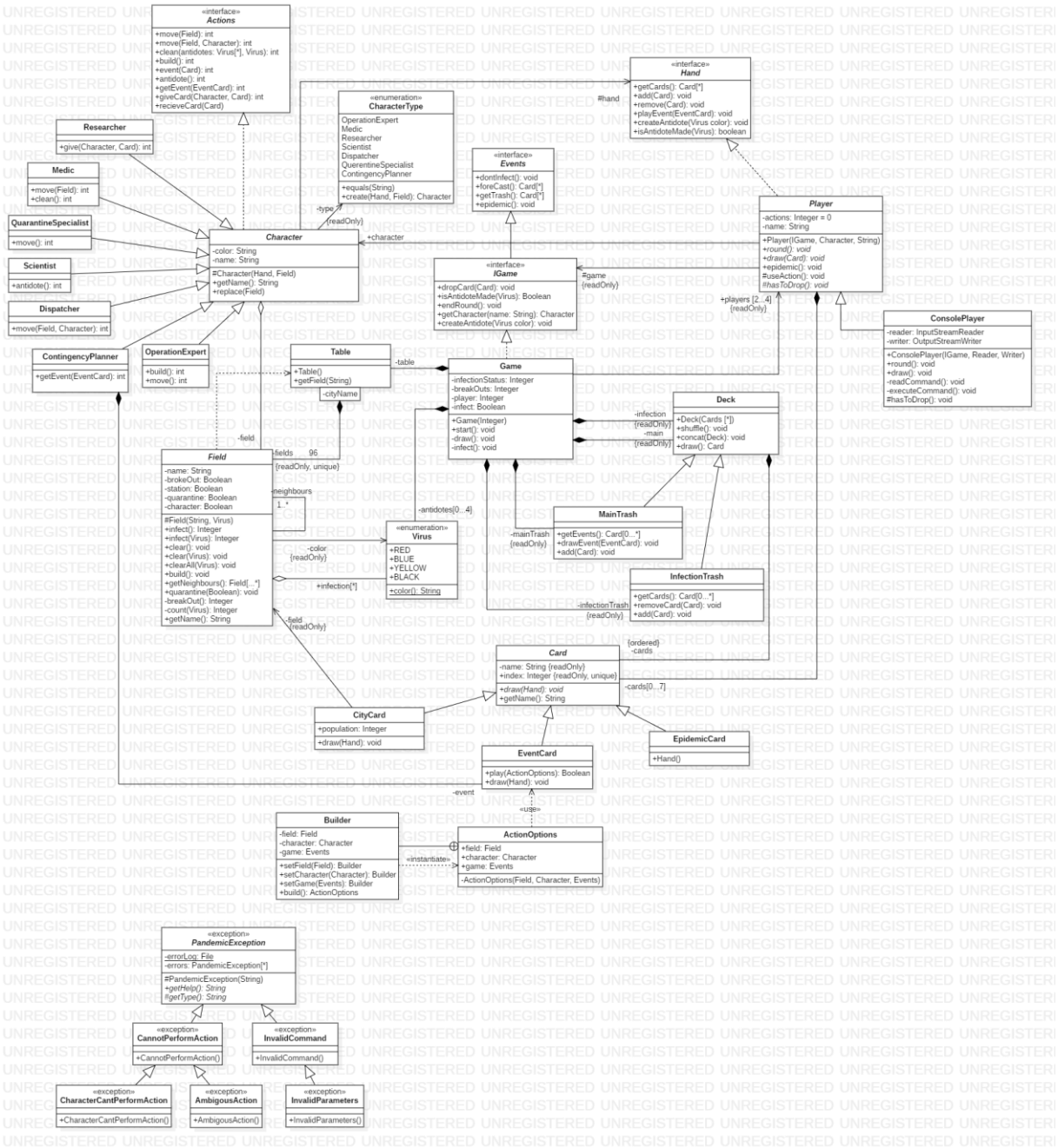
Programozás alapjai 3 házi feladat

Balassa Ádám

Osztályok



Az alábbi diagram egy áttekinthető, de pontotatlan képet ad az osztályhierarchia végleges állapotáról. Részletes és pontos (de nehezen áttekinthető) verzióját a ZIP fájlban csatoltam.

Field, Table

A Field a Table (vagy board) része, egy mező a játék táblán. Egy mező fertőzhető, állhat rajta karakter, és kutatóállomás

Character

A játék legfontosabb eleme. Egy absztrakt osztály benne minden akció default implementációja amelyre egy karakter képes. Nem példányosítható, de leszármazottjai a saját képességeiknek megfelelően a default implementáció módosítása után példányosíthatóak. Az akciók:

- Mozgás
- Más karakter mozgatása
- Tisztítás
- Kutatóállomás építése
- Ellenszer kifejlesztése
- Esemény kártya húzása

Ezen akciók mind az Action interface részei, melyet egy Character megvalósít

Player

A játékos végzi a kommunikációt a felhasználó és a játék közt. A Player egy absztrakt osztály, ennek leszármazottjai a ConsolePlayer ami egy console felületen kommunikál a user-rel és a GraphicsPlayer amely egy JavaFX grafikus felület segítségével teszi ugyanezt. Minden játékosnak van egy karaktere vannak kártyái, az ő felelőssége a saját körének megkezdése, befejezése és minden ami a sikeres kommunikációhoz szükséges.

Graphics Player

A fenti osztálydiagramon nincs feltüntetve. A Graphics Player egy állapotgép, amely a belső állapotát megváltoztatja (waitingState) amennyiben több kattintás sorozata szükséges. Ehhez tárol korábbi eseményekről és userInteractionökről adatot, melyeket töröl, amennyiben a kattintás sorozat sikeresen (vagy hibával) befejeződött. Legfontosabb függvénye az „action”, amely az aktuális állapota és a user interaction alapján meghívja a karakterének megfelelő függvényét.

Game

A Game egy osztályként tárolja a játék aktuális állapotát, ezt fájlba írva az egészében helyreállítható. A Game feleőssége a játék kezdeti állapotának beállítása (kártyák osztása, tábla létrehozása és kezdeti fertőzése) illetve a játékos egy körének vezérlése. Ő az egyetlen osztály ami hozzáfér a játékosokhoz, a paklikhoz és a táblához, amennyiben a játékos kártyát szeretne húzni, vagy eseményt kijátszani avagy a körét befejezni, ehhez az osztályhoz fordul.

Card, EventCard, CityCard, EpidemicCard

A Card egy absztrakt osztály ami egy kártyát reprezentál. 3 leszármazottja a Város kártya amely egy Field-re mutat a táblán, az Esemény kártya, amely a játék során bármikor akció nélkül kijátszható, és az Epidemic kártya amelyből csak kevés van a Main Deck-be keverve, felhúzásakor járvány tör ki. Minden kártya tudja, minek kell történnie, ha őt egy játékos kihúzza, erről értesíti a játékost. Epidemic Card felhúzása után pedig a játékos a Game-re bízza a járvány levezénylését. EventCard-ból 5db van a pakliban ezek képessége az EventCard egy belső (EventyTypes) enumerációjában van ledefiniálva. EventCard kijátszásakor annak képességétől függően meghívja a paraméterként kapott játékos megfelelő metódusait, egy „opciók” paraméterben kapott megfelelő változókkal. Az opciók (EventOptions) szintén egy belső osztálya az EventCard-nak, tartozik hozzá egy builder, amivel a megfelelő mezői beállíthatók.

Deck

A kártyák paklikban vannak, ezek lehetnek dobópaklik (random access) és húzó paklik, melyeknek csak a tetejéről lehet húzni lapot. Egyes paklik keverhetők, egyes pakliknak az utolsó lapja is

kihúzható, mások összefűzhetők más paklikkal, ezek a Deck (mint absztrakt osztály) leszármazottjaiban vannak definiálva

Pandemic

Az alkalmazás belépési pontja, egyben a JavaFX Application. Ő felelős a játék felkonfigurálásáért illetve elindításáért. Feldob egy ablakot amelyben ki lehet választani a szükséges beállításokat (kezelőfelület, játékosok száma, nehézség, új játék/ korábbi játék betöltése). A beállítások alapján elindítja a játékot a megfelelő kezelőfelülettel. Mivel ez az osztály felelős korábbi játék esetleges betöltéséért, a szerializálást ő végzi.

PandemicException

Az alkalmazás custom-exceptionöket is definiál amik akkor történnek, ha a felhasználó szabálytalan akciót hajtana végre. Leszármazottjai a CannotPerformAction, amely egy általános exception valamilyen illegal action-re, ennek speciális esete, amikor a user olyan akciót hajtana végre ami nem okozna változást a játék állapotán, de felhasználna egy akciót (UnnecessaryAction) illetve a CharacterCannotPerformAction, amikor az akció azért nem hajtható végre mert erre még a karakter is képtelen volna. Az AmbigousAction egy olyan CannotPerformActionException, melynek jelentése, hogy az akció két ugyanannyi erőforrást felhasználó végreahjtási módja van, a felhasználó feladata a választás. Van ezen kívül InvalidCommand és InvalidParameter exception, amelyek belső hibák, hibátlan működés esetén nem szabadna megtörténniük. Van ezen kívül EndOfGame, amennyiben a játékosok vesztettek illetve EndOfRound amely akkor történik, ha a a játékosnak elfogytak az akciói, de nem passzolt.

Grafika

A grafikához JavaFX-et használtam. Minden megjelenített elemhez külön grafikus komponenst hoztam létre, melyet a 2 Scene felhelyezhet a Stage-re. A MainScene a konfigurációs felülete a játéknak, a GameScene pedig a játék tényleges felülete. A komponensek legjelentősbb részét a GameScene használja. Majdnem minden komponens frissíthető (refresh), ennek meghívásakor, az újrarenderelődik, illetve a GameScene maga is frissíthető, amely didaktikusan frissít minden komponenst. A GameScene controllere maga a GraphicsPlayer, vagyis nekik kölcsönösen van referenciájuk egymásra. Ezen kívül egyes komponensek, mint a ControllerComponent (alsó gombok) Field (mező) stb. külön is tárolnak referenciát az aktuálisan körön lévő játékosra, ezek elemeire való kattintáskor ők közvetlenül értesítik a GraphicsPlayer-t (meghívják annak action függvényét). Természetesen új kör kezdetekor minden ilyen komponensnek is frissítenie kell az aktív játékosra való referenciáját, ez a GameScene felelőssége.

Definiáltam ezen kívül saját animációkat illetve gyakran használt függvény sorozatokat, ezeket az Effects osztályban foglaltam össze, melynek csak statikus függvényei vannak.

A grafikát megvalósító osztályaim teljesen el vannak szeparálva a logikáért felelősekkel, ez a fájlstruktúrában is meglátszik. A szükséges képeket, ikonokat egy, az src-vel egy szinten lévő res mappában definiáltam.

A dokumentáció a grafikus felületet ennél részletesebben nem tárgyalja, minden függvényem csak JavaFX panelek létrehozásával illetve event-ek állításával foglalkozik, nevük többnyire egyértelműen definiálja működésüket, minden komponens egy init és egy (vagy több) refresh függvénnyel rendelkezik, az init a létrehozáskor, a refresh a frissítés szükségességekor hívódik meg.

Függvények

A házifeladatom 42 osztályból és közel 4200 sorból áll (kommentek és üres sorok nélkül) így minden függvényre és osztályra a dokumentációban nem térek ki. A legfontosabb függvények működését JavaDoc-kal dokumentáltam, ez megtekinthető a feltöltött kódban, illetve a kiexportált hypertext-ként az alábbi github repository-ben:

https://github.com/bizmut32/pandemic-documentation