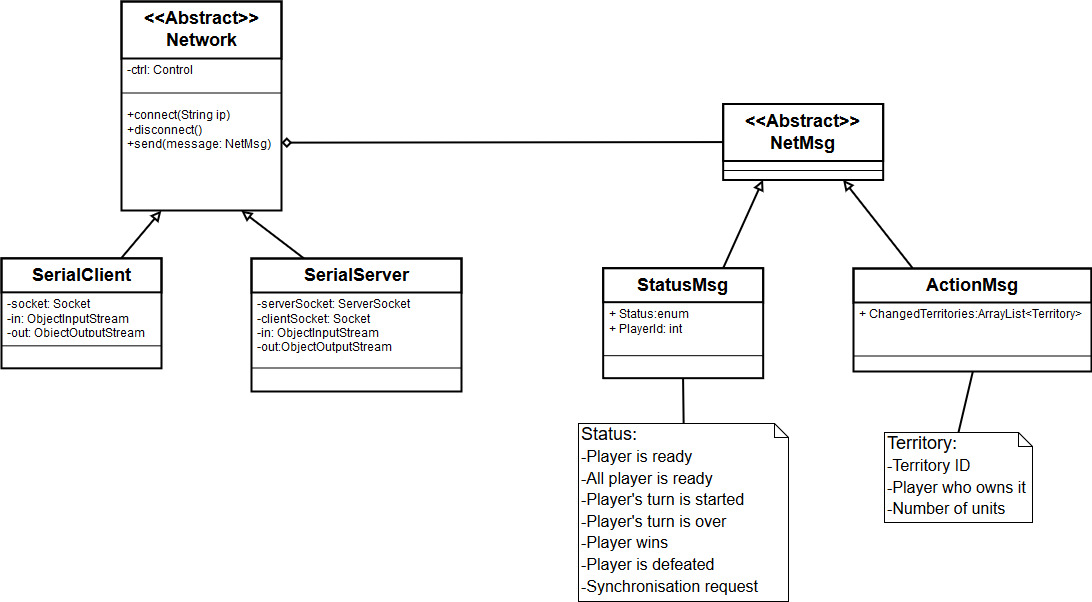
# Osztálydiagramok

# GUI osztálydiagram

# 

## Network Osztálydiagram

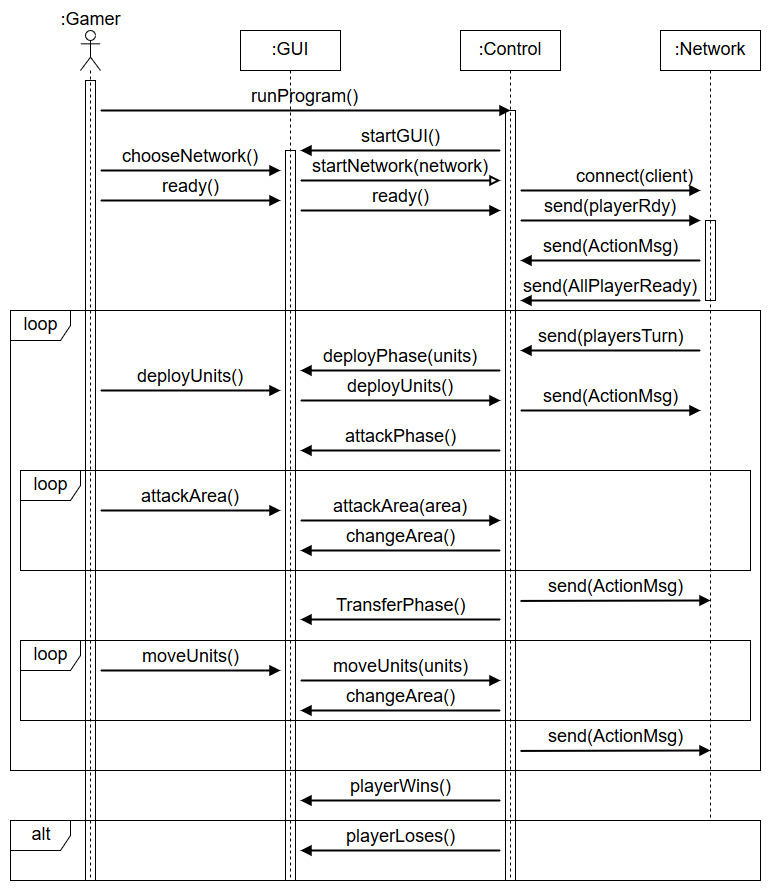


A hálózat osztálydiagramja hasonló az órai példában lévő osztálydiagramhoz, vagyis itt is egy absztrakt Network osztály van, amiből leszármazik a SerialClient és a SerialServer. Mind a két osztály örökli a Network tulajdonságait és metódusait. Amit kiemelnék az a „send()” metódus. A send metódusban csak NetMsg osztályokat lehet küldeni.  
A NetMsg egy absztrakt osztály, belőle származnak a StatusMsg és az ActionMsg osztályok.  
A StatusMsg osztály arra szolgál, hogy jelzéseket lehessen küldeni a szerver és a kliensek között. Ezek az üzenetek a játékos azonosítójából állnak, illetve egy enum típusú jelzésből, ezek readonly tulajdonságok, csak az üzenet létrehozásakor adhatók meg.  
A jelzés lehet:

* Ready: A játékos ezzel jelzi, hogy készen áll elkezdeni a játékot
* AllPlayerReady: A szerver ezzel jelzi a klienseknek, hogy kezdetét veheti a játék
* TurnStarted: Jelzi, ha a játékos köre elkezdődött
* TurnOver: Jelzi, ha a játékos köre véget ért
* Win: Jelzi, ha a játékos nyert
* Defeat: Jelzi, ha a játékos veszített
* Sync: (későbbi implementációra) A játékos ezzel jelzi, hogy kiesett a térkép szinkronizációjából(pl. timeout miatt), ekkor a szerver elküldi neki a teljes világtérképet

Az ActionMsg osztály tartalmazza a megváltozott területeket. Ezzel lehet jelezni pl. egy támadás kimenetelét a szerver felé. A ChangedTerritories egy ArrayList típusú tulajdonság, nem fix méretű tároló, tehát adott esetben az összes területet el lehet küldeni egy ActionMsg üzenetben.

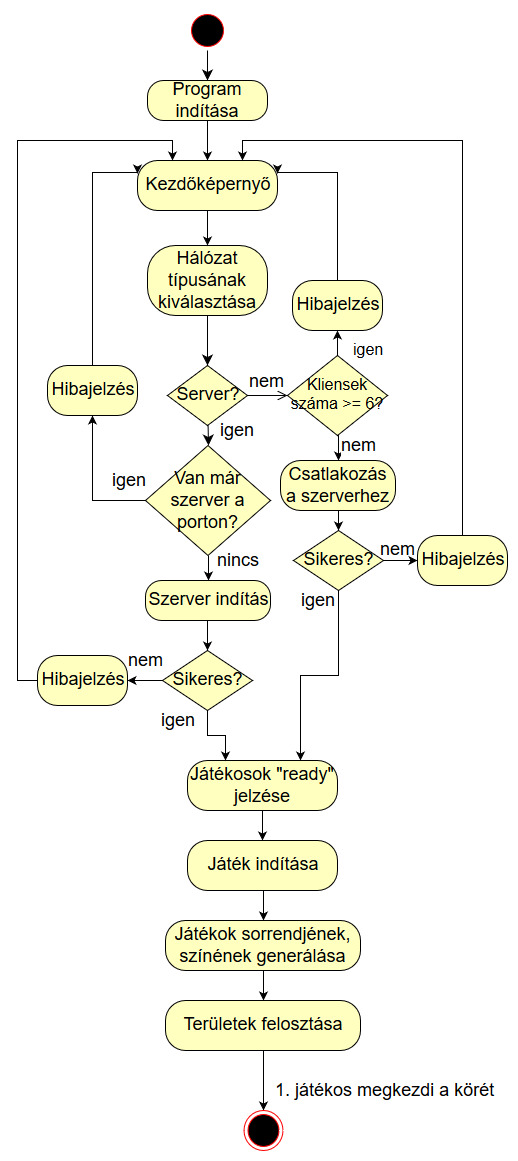
Szekvencia Diagram



A szekvencia diagramon a játékos elindítja a játékot, majd az megjeleníti a grafikus felületet. A játékos ezután kiválasztja, hogy a hálózat típusát, ami lehet szerver vagy kliens, és jelzi, hogy készen áll a játékra(ready jelzés). Ha az összes csatlakozott játékos készen áll, akkor a szerver inicializálja a világtérképet, elküldi a klienseknek a kiosztott területeket, és elküldi a klienseknek, hogy kezdetét veszi a játék. Ezután a játék egy iteratívvá válik (első loop frame), aminek a játékos szempontjából két kimenet lehetséges, a játékos vagy nyer, vagy veszít. Amíg ezek valamelyike nem következik be, addig a játékos várakozik, hogy az ő köre következzen, majd ha a szerver jelez, hogy ő következik, akkor elkezdi a körét.  
A játékos körének első fázisa a „Deploy” fázis. Ebben a fázisban a Control kiszámolja, hogy a területei alapján mennyi egységet helyezhet el. A játékos elhelyezi ezeket az egységeket, majd a fázis végén a kliens jelzi a szervernek a megváltozott területeket.  
A szerver ezután jelzi a játékosnak, hogy a következő fázisa, az „Attack” fázis következik. Ekkor a játékos megadhat egy támadást. A játékban lezajlik a harc, majd a játék visszajelez a grafikus felületen a harc kimeneteléről. A játékos ezután dönthet úgy, hogy folytatja a harcot vagy sem. Ha nem akkor a kliens értesíti a szervert a megváltozott területekről (tehát nem minden támadás után történik hálózati üzenetküldés, csak a fázis végén).  
A szerver jelzi, hogy a játékosnak a „Transfer” fázisa következik. Ebben a fázisban a játékos egységeket mozgathat, az előző fázisban leírtak szerint (a harc nélkül természetesen, de a játék mechanika ugyanaz).

# Aktivitás Diagram

## A játék indulásának aktivitás diagramja



## A játékos köreinek aktivitás diagramja

