Potyogós

Szoftvertechnológia házi feladat

Rittgasszer Ákos

Z8WK8D

1 A FELADAT LEÍRÁSA

A játékot N játékos játssza (N >= 2, de tetszőlegesen nagy egész szám lehet). A játék egy N oldalú függőleges táblából áll, melybe forgatható tárcsák vannak beépítve. A tárcsák a tábla minden oldalán látszanak. Ha egy tárcsa forog, a tárcsa minden oldala együtt forog. A tárcsák szélén kis zsebek vannak. Egy tárcsa egyes oldalain a zsebek száma, illetve az zsebek pozíciója különböző is lehet.

Minden játékos néhány számozott koronggal (érmével) kezd a tábla tetején. Egy kis zsebbe pontosan egy korong illeszkedik. Ha két érintkező tárcsa zsebei egymás mellé kerülnek, a felső zsebben lévő korong átpottyan az alsó zsebbe.

A játék célja az érmék eljuttatása a tábla tetejéről a tábla alján lévő tálcába a tárcsák forgatásával. A játékosok felváltva forgatnak egy-egy tárcsát, de az a tárcsa nem mozgatható, amelyet az előző játékos éppen mozgatott. Amíg a játékos soron van, a kiválasztott tárcsát bármilyen pozícióba forgathatja, akár több irányba is és akár többször is. A győztes az a játékos, akinek először pottyan le az összes érméje a tálcába. A korongoknak a számozás sorrendjében kell leérkezniük.

Mivel egyik játékos sem látja a többi játékos tábláját, gyakran előfordulhat, hogy segítik vagy éppen hátráltatják a többiek előrehaladását. A játék arra késztet, hogy előre tervezzünk: belehajszolhatjuk a többi játékost, hogy a saját tárcsája forgatásával a mi korongjainkat vigye tovább, de közben vigyázzunk arra, nehogy rossz sorrendben potyogjanak le a korongjaink a tálcára.

2 FUNKCIONÁLIS KÖVETELMÉNYEK

2.1 ELSŐDLEGES KÖVETELMÉNYEK

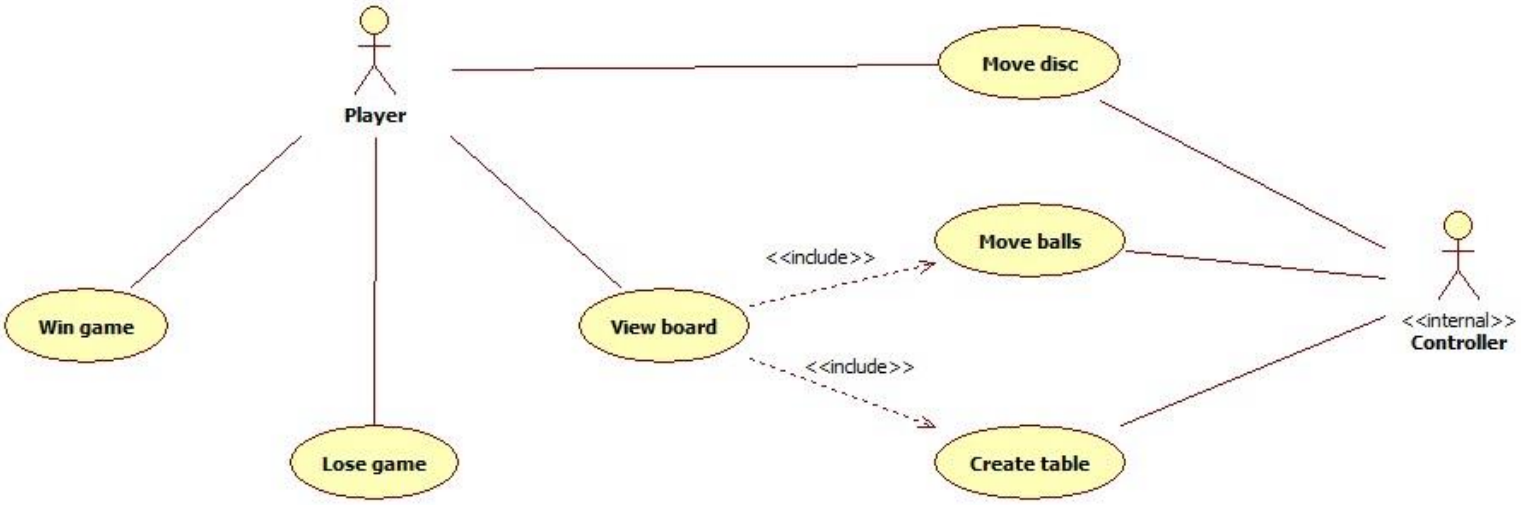
Azonosító	**Leírás**	**Use-case**
R01	A játék egy N oldalú függőleges táblából áll, melybe forgatható tárcsák vannak beépítve	View board, Move disc
R02	A tárcsák a tábla minden oldalán látszanak	View board
R03	Ha egy tárcsa forog, a tárcsa minden oldala együtt forog	Move disc
R04	A tárcsákon zsebek vannak, számuk és helyzetük különböző	View board, Create game
R05	Ha két érintkező tárcsa zsebei egymás mellé kerülnek, a felső zsebben lévő korong átpottyan az alsó zsebbe	Move balls
R06	A játékosok felváltva forgatnak egy-egy tárcsát	Move disc
R07	Az a tárcsa nem mozgatható, amelyet az előző játékos éppen mozgatott	Move disc
R08	Amíg a játékos soron van, a kiválasztott tárcsát bármilyen pozícióba forgathatja, akár több irányba is és akár többször is	Move disc
R09	A korong belép a játékba, bekerül a kiindulási tárolóból az első zsebbe	Move balls
R10	A korong kilép a játékból, bekerül a végső tárolóba az utolsó zsebből	Move balls
R11	Mindegyik táblán ugyanúgy vannak a zsebek	Creat game

2.2 TOVÁBBI KÖVETELMÉNYEK

Azonosító	Leírás	Use-case
R12	Minden tárcsán van legalább egy zseb	Create game
R13	Ha rossz számú korong esik le alura a játékos kiesik	Move balls, Lose game
R14	Ha minden korióong lent van jó sorrendben a játékos nyer (egy nyertes van)	Move balls, Win game

3 USE-CASE-EK

3.1 USE-CASE DIAGRAM



3.2 USE-CASE LEÍRÁSOK

Cím	Move disc
Leírás	A játékos tudják a saját körükben mozgatni a korongot, a controller pedig a játékos által mozgatott koronggal szinkronban mozgatja a többi játékosét
Aktorok	Player, Controller
Főforgatókönyv	1. A játékos mozgat egy kiválasztott korongot a köre végéig 2. A controller mozgatja a többi játékos korongját szinkronban 3. A korong zsebeibe kis és be tudnak esni a számozott labdák
Alternatív forgatókönyv	1. Ha azt a korongot akarja mozgatni a játékos amit az előző körben mozgatott az előző játékos akkor az nem lehetséges

Cím	**View board**
Leírás	A játékos megnézi a saját tábláját (többiekét nem látja)
Aktorok	Player
Főforgatókönyv	1. Kirajzolódik a tábla aktuális állapota 2. A játékos megnézi a saját táblája aktuális állapotát

Cím	**Move Balls**
Leírás	A labdák mozognak a korongok zsebei között
Aktorok	Controller
Főforgatókönyv	1. Ha két zseb egymás mellé ér akkor a fentebbiből átesik a lentebbibe a labda
Alternatív forgatókönyv	1. A legfelső korong zsebeibe egy tárolóból (amiben sorban vannak a labdák) tud beleesni 2. A legalsó korong zsebeiből kiesve a labda kiesik a játékból: 1. Ha a kieső labda száma ez előző kiesőnél eggyel nagyobb (és az 1-es labda esett ki először) akkor a játékos közelebb kerül a győzelemhez 2. Ha a kiső labda száma nem eggyel nagyobb az előtte kiesőénél, vagy nem az 1-es számú labda esik ki először, akkor a játékos vesztett

Cím	**Create game**
Leírás	A controller legenerálja a táblákat véletlenszerűen a játék elején
Aktorok	Controller
Főforgatókönyv	Legenerálja táblákat, mindegyik játékosnál ugyanúgy helyezkednek el a korongok, csak a zsebek helyzete eltérő

Cím	**Lose game**
Leírás	Veszít a játékos, nem a megfelelő korongja esik le
Aktorok	Player
Főforgatókönyv	Kiesik a játékból

Cím	**Win game**
Leírás	Nyer a játékos, lent van minden korongja
Aktorok	Player
Főforgatókönyv	Vége a játéknak, megvan a nyertes

4 STRUKTURÁLIS LEÍRÁS

4.1 AZ OSZTÁLYOK LEÍRÁSA

4.1.1 Game

Felelősségek

Végigviszi a játékot. Legenerálja a táblákat, aztán pedig addig mennek a körök amíg vége nem lesz a játéknak. A végén kiírja az eredményt.

Attribútumok

-boards: Board[0…*]	A játékhoz tartozó táblák

Metódusok

+Init()	A játék kezdetén hívódik meg és legenerálja a játékosok számának megfelelő számú táblát. Megmondja mindegyik tárcsáról, hogy melyik tárcsákkal szomszédos (board.neighbourDiscs) és, hogy mi az indexe (storageIdx). Minden táblán ugyanúgy helyezkednek el a tárcsák, csak a zsebek elhelyezkedése eltérő. A felső tárolónak 0, az alsónak a tárcsák száma plusz 1, közte pedig a tárcsáké fentről lefelé nő
+Play(): void	A tényleges játékot halytja végre. Addig játszanak a játékosok a táblákon amíg nincsen vége a játéknak. Eltárolja az előzőleg mozgatott tárcsát. Ellenőrzi az esetleges nyertest és a veszteseket
+End(winner: Gamer): void	Ha vége van a játéknak a Play() meghívja az End() függvényt. Ennek paramétere a nyertes játékos

4.1.2 Board

Felelősségek

A táblákon lehet végrehajtani az aktuális lépést. A lépés figyelembe veszi, hogy nem lehet az előzőleg forgatott korongot forgatni. A táblákhoz hozzá van rendelve a játékos akinek a tábla a játéktere. A táblához tartozik a felső és az alsó korongtároló is.

Attribútumok

-gamer: Gamer	Egy táblához egy játékos tartozik. Ennek a játékosnak a körében lehet módosítani a táblát.
-upStorage: UpStorage	A labdák itt vannak alapból és innem tudnak bekerülni a játékba, az storageIdx-e 0
-downStorage: DownStorage	Ide esnek le a labdák ha kiesnek az utolsó korongból. Ha kiesnek akkor a CheckLose() ellenőrzi, hogy jó-e a sorrend. Ha már minden labda kiesett, akkor a CheckWin() mondja meg, hogy jó sorrendben végzett-e. A labdák itt vannak alapból és innem tudnak bekerülni a játékba, az storageIdx-e tárcsák száma plusz 1
-discs: Disc[1…*]	Ebben a tömbben vannak a táblához tartozó tárcsák.

Metódusok

+Step(movedDiscIdx int): int	Egy lépés végrehajtásáért felelős. A ChoseDiskToMove függvény által kiválasztott tárcsára meghívja annak a Move metódusát. Visszatér az mozgatott korong indexével. Ez mozgatja az összes többi tábla azon tárcsáját amit a
+CheckWin(): bool	Visszatér egy logikai változóval, hogy van vagy nincs nyertes
+CheckLose(): bool	Visszatér, hogy vesztet-e az aktuális tábla játékosa
+RomoveCurrentBoard()	Eltávolítja az aktuális táblát a játékban levő táblák közül
-ChoseDiscToMove (movedDiscIdx: int): Disc	Kiválasztja az alapján, hogy mejik tárcsa volt mozgatva az előbb, hogy melyik less ebben a körben mozgatva és visszatér vele

4.1.3 Gamer

Felelősségek

Ez az osztály képviseli az egyes játékosokat amiket hozzárendelünk táblákhoz.

Attribútumok

+name: string	Szöveg, ami a játékosok megkülönböztetésére alkalmas

4.1.4 Ball

Felelősségek

Ez az osztály képviseli a korongokat. Mindegyik korongnak avan egy száma, ami fontos a játékhoz. Ez az osztály kezeli azt, hogy a korongok átesnek az egyik helyről a másikba.

Attribútumok

-number: int	A korongok száma, ami fontos a játékhoz

4.1.5 Disc

Felelősségek

Az eggyes tárcsák múködéséért felelős osztály. A BallStorage osztály leszármazotja. Annyi korongot tárol aminnyi zseb van rajta. Indexel lehet hivatkozni az egyes zsebekre

Attribútumok

-neighbourDiscs: Disc[0...*]	Egy tömb, amiben benne vannak a szomszédos tárcsák és a felső és also tárolók ha valamelyikükkel szomszédos. Ez egy heterogén kollekció
-degreeOfPockets	A tárcsán levő zsebek szögét tárolja a függőlegeshez képest

Metódusok

+Move(): void	Forgatja a tárcsát, ha talál a szomszédos tárcsák (vagy a felső illetve also tároló) között olyan zsebet amiből át tud esni vagy amibe be tud esni a labda (ami a megfelelő zsebben van), akkor az átesik. A FallIn, FallOut és Fall függvényeket használja. Forgatja a
-FallIn(neighbourDisc: Disc[0...*]): BallStorage	Visszatér azzal a tárolóval ahonnan be tud esni az aktuális tárcsába korong. Az indexek és szögek segítségével dönti el mikor van átesés
-FallOut(neighbourDisc: Disc[0...*]): BallStorage	Visszatér azzal a tárolóval ahova be tud esni az aktuális tárcsából korong. Az indexek és szögek segítségével dönti el mikor van átesés
+Fall(to: BallStorage, idx: int): void	A FallIn és FallOut függvények visszatérési értékei segítségével végrehajtja a korongmozgásokat. Abban különbözik a szülője azonso nevű függvényétől, hogy meg lehet mondani, hogy hova rakja a korongot (a tárcs melyil zsebébe)

4.1.6 BallStorage

Felelősségek

A korongok tárolására alkalmas abstract osztály, az új adatot a tömb végére teszi be

Attribútumok

#numOfBalls: int	Az aktuálisan a tárolóban levő korongok száma
-balls: Ball[0…*]	A tárolóban levő korongok tömbje
-storageIdx: int	Az init függvényben kerül megadásra. Ennek segítségével esnek le vagy sem a korongok

Metódusok

#AddBall(ball: Ball): void	Hozzád egy korongot a tárolóhoz, a végére rakja be
#RemoveBall(ball: Ball): void	Eltávolítja a paraméterül kapott elemet a tárolóból
+Fall(to: BallStorage): void	Az aktuális tárolóból atesik a korong ami át tud esni (ha van ilyen) a paraméterül kapottba

4.1.7 DownStorage

Felelősségek

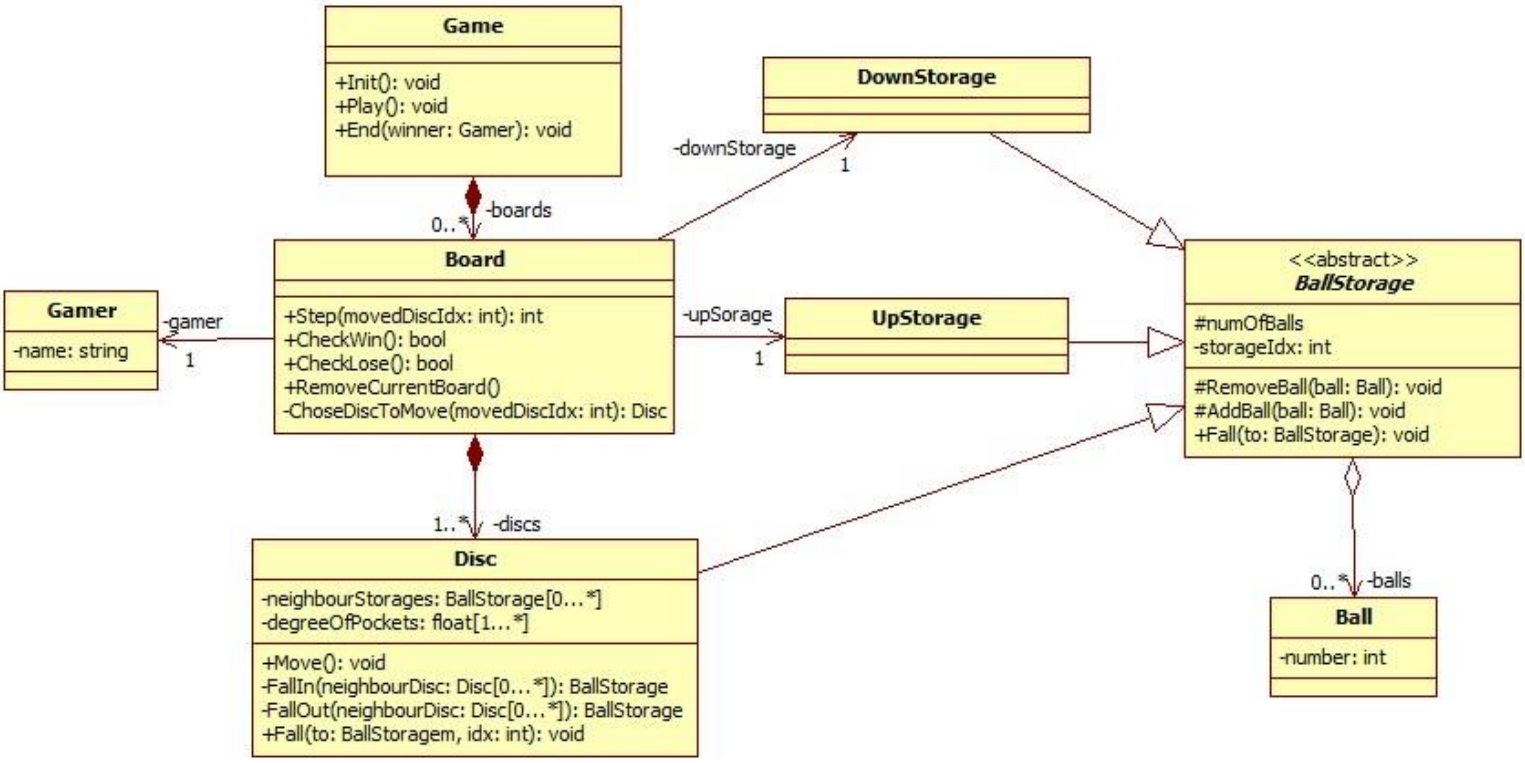
A BallStorage osztályból származik le, működése megeggyezik a szülőjével. Azokat a korongokat tárolja amik kiestek a táblából. Be tud esni bele labda de ki nem, ezért indexe fixen egyel több mint a tárcsák száma.

4.1.8 UpStorage

Felelősségek

A BallStorage osztályból származik le, működése megeggyezik a szülőjével. Azokat a korongokat tárolja amik még nincsenek játékba. Ki tud esni belőle korong de be nem, ezért indexe fixen 0.

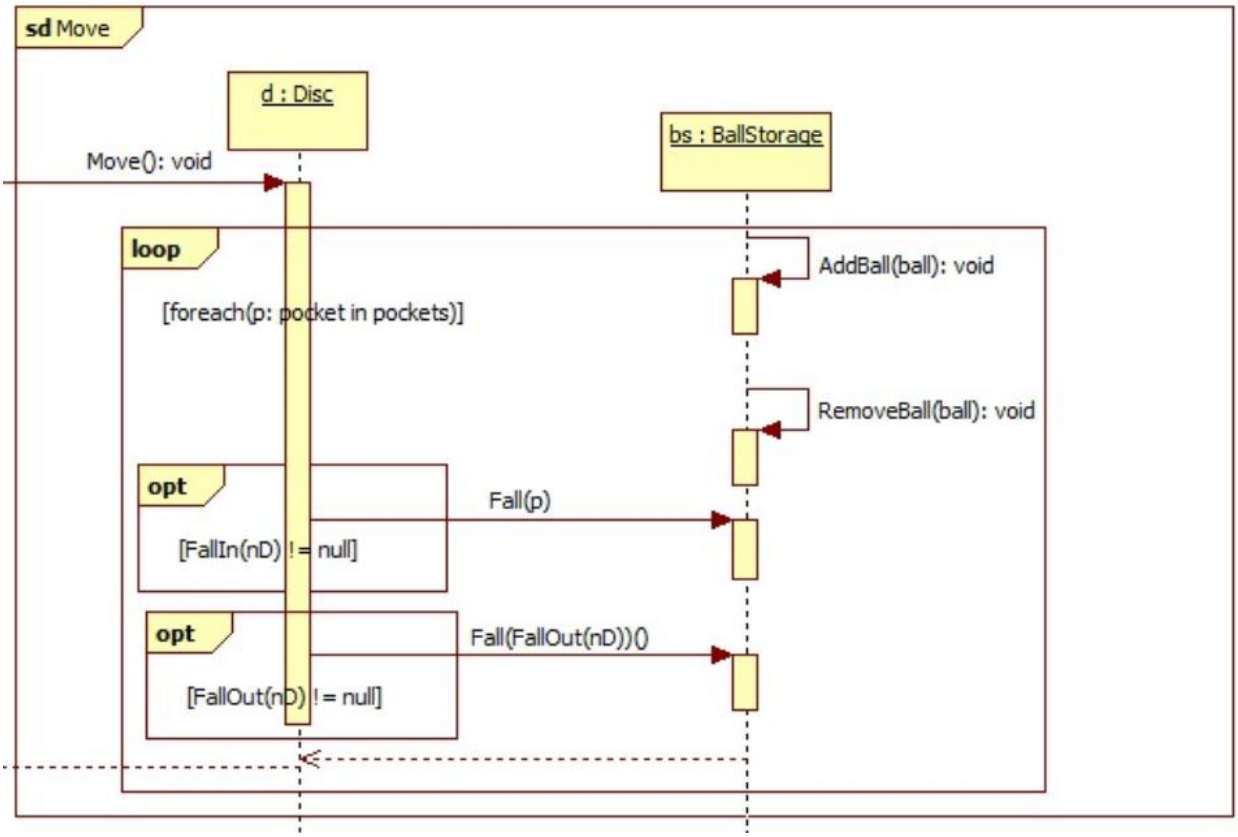
OSZTÁLYDIAGRAM



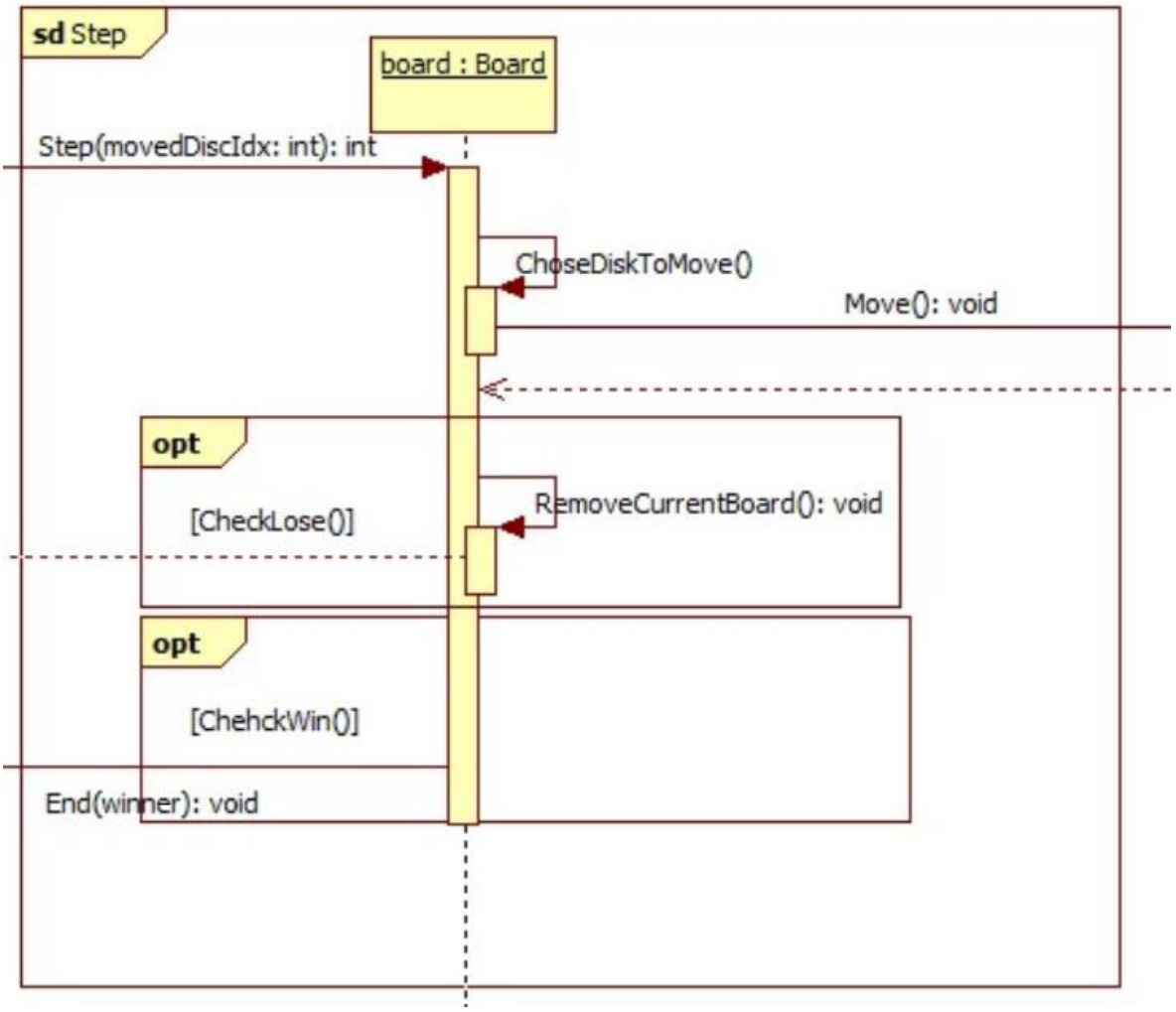
5 VISELKEDÉS LEÍRÁSA

5.1 SZEKVENCIA DIAGRAMOK

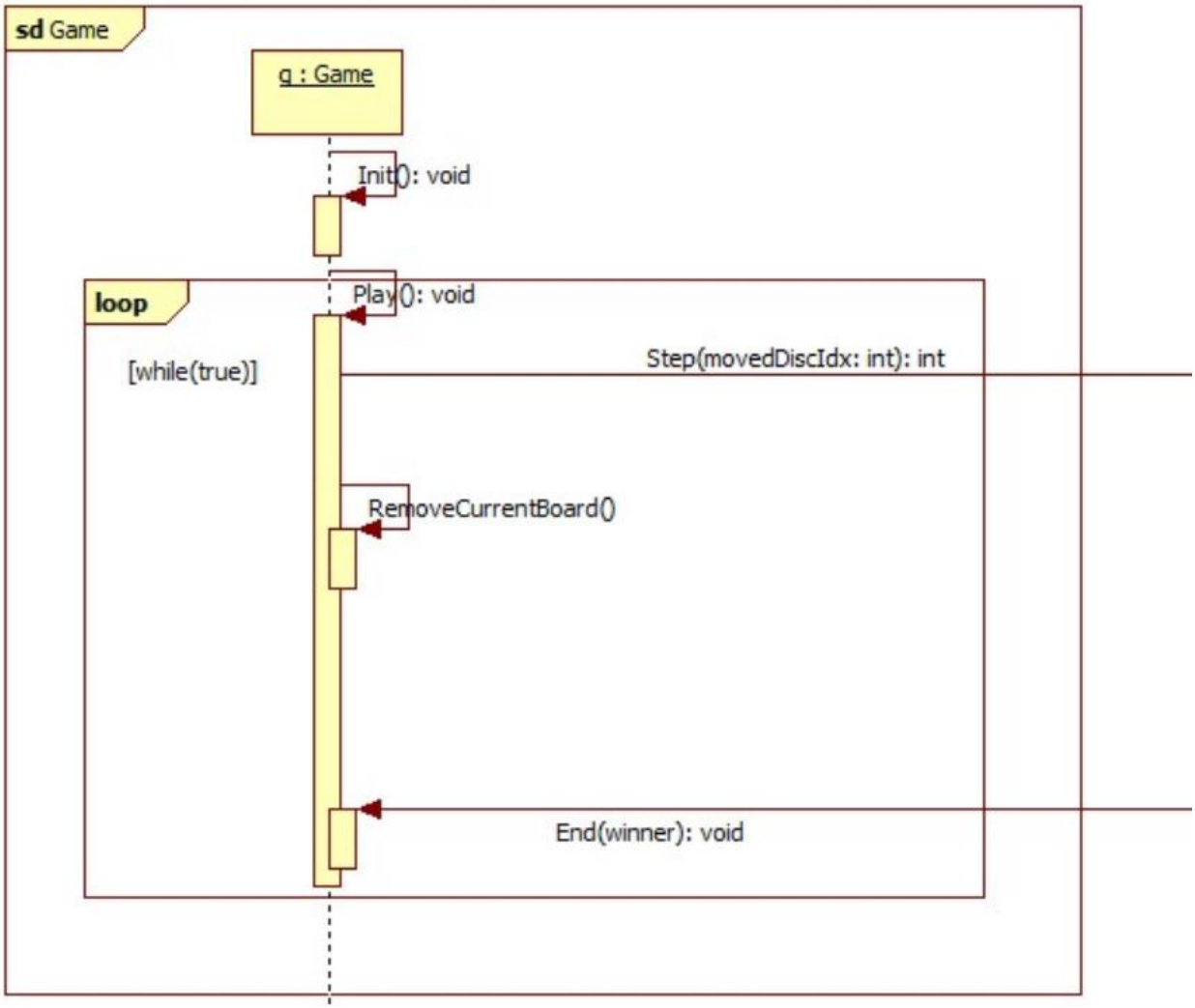
5.1.1 Move



5.1.2 Step


sd Step
board : Board
Step(movedDiscIdx: int): int
ChoseDiskToMove()
Move(): void
opt
[CheckLose()]
RemoveCurrentBoard(): void
opt
[ChehckWin()]
End(winner): void

5.1.3 Game


sd Game
g : Game
Init(): void
loop
Play(): void
[while(true)]
Step(movedDiscIdx: int): int
RemoveCurrentBoard()
End(winner): void

6 NAPLÓ

Kezdet	Időtartam	Elvégzett munka	Hivatkozások
2019.10.23. 15:00	1.5 óra	Követelmények felsorolása, use-case-ek összegyüjtése és a diagram elkészítése	2, 3
2019.10.26. 11:00	4 óra	Osztályok leírása, osztálydiagramm elkészítése	4
2019.11.03. 16:00	2 óra	Osztályok újragondolása, osztálydiagramm korrigálása	4
2019.11.6. 20:00	4 óra	Szekvenciadiagramm megszerkesztése	5
2019.11.9. 17:00	5 óra	Az egész újra átgondolása és kisebb módosítása	2, 3, 4, 5
2019.11.10. 18:00	1 óra	Átnézés, kisebb hibák javítása. Dokumentáció véglegesítése	2, 3, 4, 5

Összes elvégzett munka:17.5 óra

Modellező eszköz: WhiteStarUML

Egyéb eszközök: Microsoft Word