Programozási technológia
1. Beadandó

Név: Magyar Viktor

Neptun kód: O9WEJJ

Dátum: 2024.02.25.

Feladat száma: 3

1. Feladat leírása:

Szimuláljuk az alábbi egyszerűsített Capitaly társasjátékot! Adott néhány eltérő stratégiájú játékos és egy körpálya, amelyen különféle mezők sorakoznak egymás után. A pályát körbe-körbe újra és újra bejárják a játékosok úgy, hogy egy kockával dobva mindig annyit lépnek, amennyit a kocka mutat. A mezők három félék lehetnek: ingatlanok, szolgáltatások és szerencse mezők. Az ingatlant meg lehet vásárolni 1000 Petákért, majd újra rálépve házat is lehet rá építeni 4000 Petákért. Ha ezután más játékos erre a mezőre lép, akkor a mező tulajdonosának fizet: ha még nincs rajta ház, akkor 500 Petákot, ha van rajta ház, akkor 2000 Petákot. A szolgáltatás mezőre lépve a banknak kell befizetni a mező paramétereként megadott összeget. A szerencse mezőre lépve a mező paramétereként megadott összegű pénzt kap a játékos. Háromféle stratégiájú játékos vesz részt a játékban.

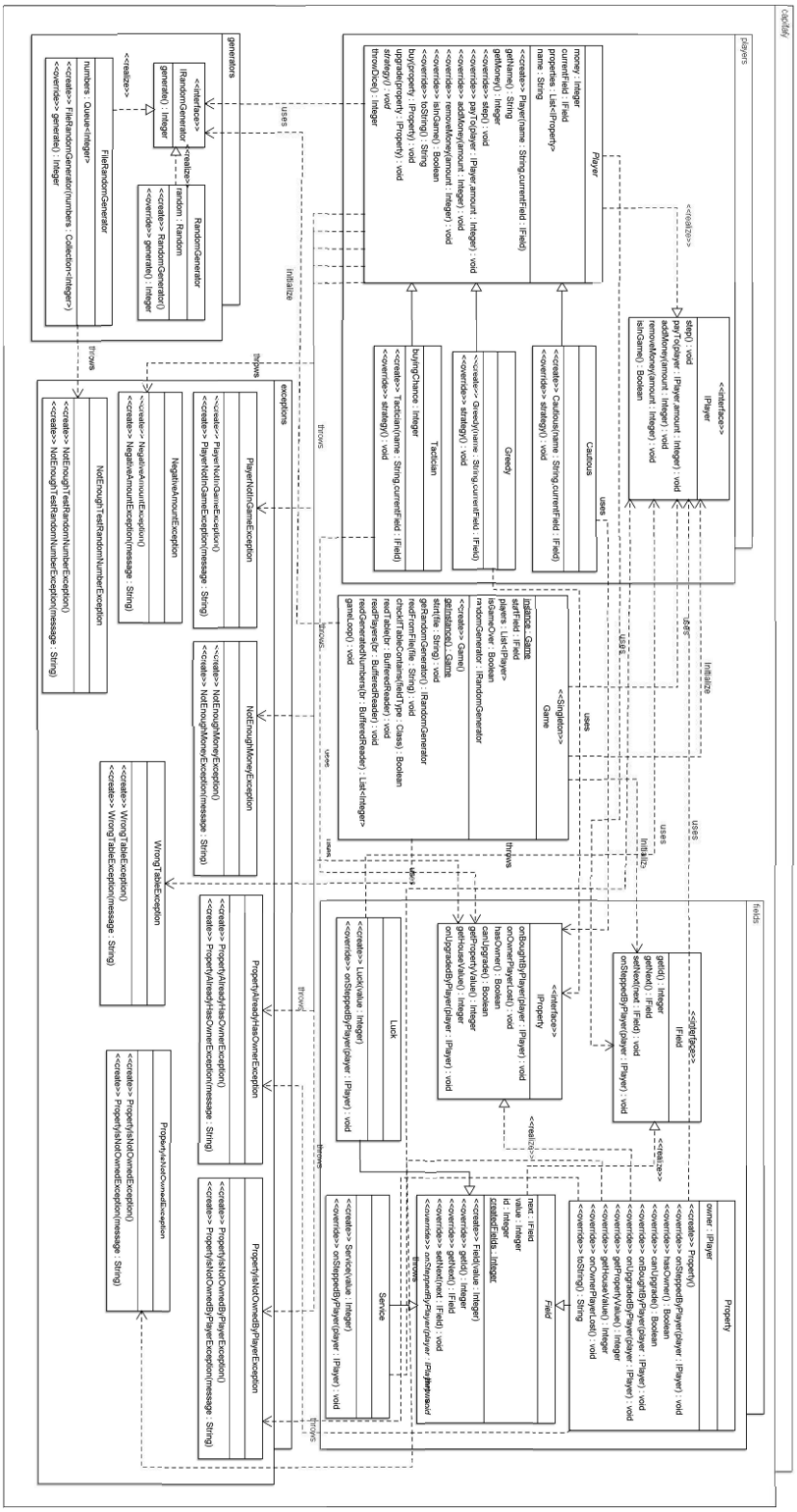
Kezdetben mindenki kap egy induló tőkét (10000 Peták), majd a „mohó” játékos ha egy még gazdátlan ingatlan mezőjére lépett, vagy övé az ingatlan, de még nincs rajta ház, továbbá van elég tőkéje, akkor vásárol. Az „óvatos” játékos egy körben csak a tőkéjének a felét vásárolja el, a „taktikus” játékos minden második vásárlási lehetőséget kihagyja. Ha egy játékosnak fizetnie kell, de nincs elégendő pénze, akkor kiesik a játékból, házai elvesznek, ingatlanjai megvásárolhatókká válnak.

A játék paramétereit egy szövegfájlból olvassuk be. Ez megadja a pálya hosszát, majd a pálya egyes mezőit. Minden mezőről megadjuk annak típusát, illetve ha szolgáltatás vagy szerencse mező, akkor annak pénzdíját. Ezt követően a fájl megmutatja a játékosok számát, majd sorban minden játékos nevét és stratégiáját. A tesztelhetőséghez fel kell készíteni a megoldó programot olyan szövegfájl feldolgozására is, amely előre rögzített módon tartalmazza a kockadobások eredményét.

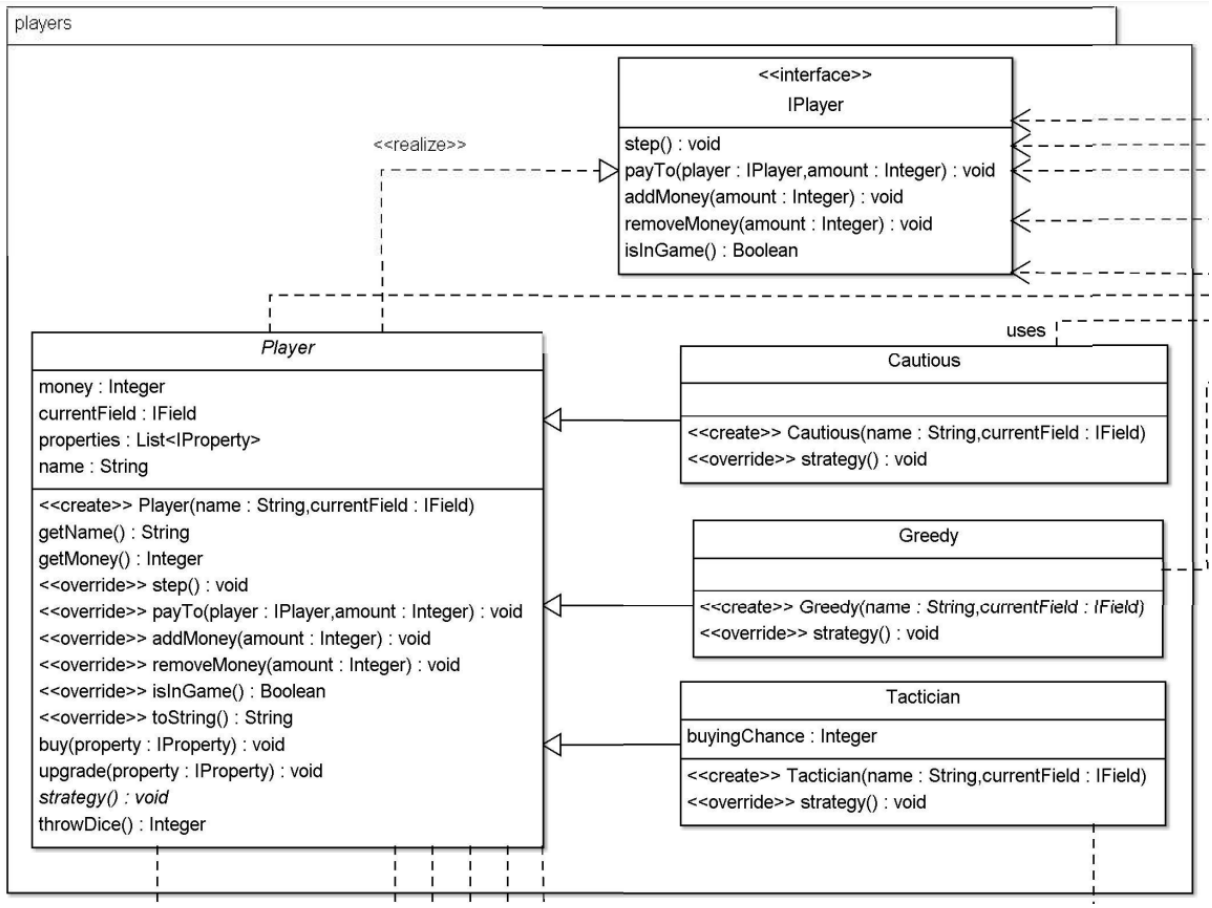
Írjuk ki, hogy adott számú kör után hogyan állnak (mennyi a tőkéjük, milyen ingatlanokat birtokolnak) a versenyzők!

2. Megoldási terv:

2.1 Teljes osztálydiagram:



2.2 capitaly.players package osztályok leírása:



2.2.1 IPlayer:

Egy interfész, ami egy játékos publikus metódusait írja le. Minden más osztály, ami nem ebben a csomagban van, ezt az interfészt használja.

2.2.2 Player:

Egy absztrakt osztály egy játékos reprezentálására. Implementálja az IPlayer interfész metódusait, illetve egyéb protected metódusokkal bír. A leszármazott osztályokban a strategy absztrakt metódus implementálásában kell megadni az adott játékos logikáját, hogy mi alapján lép. Ez a strategy metódus a step metódusban hívódik meg.

2.2.3 Cautious:

A strategy metódust implementálja, amiben az „óvatos” játékos logikája van.

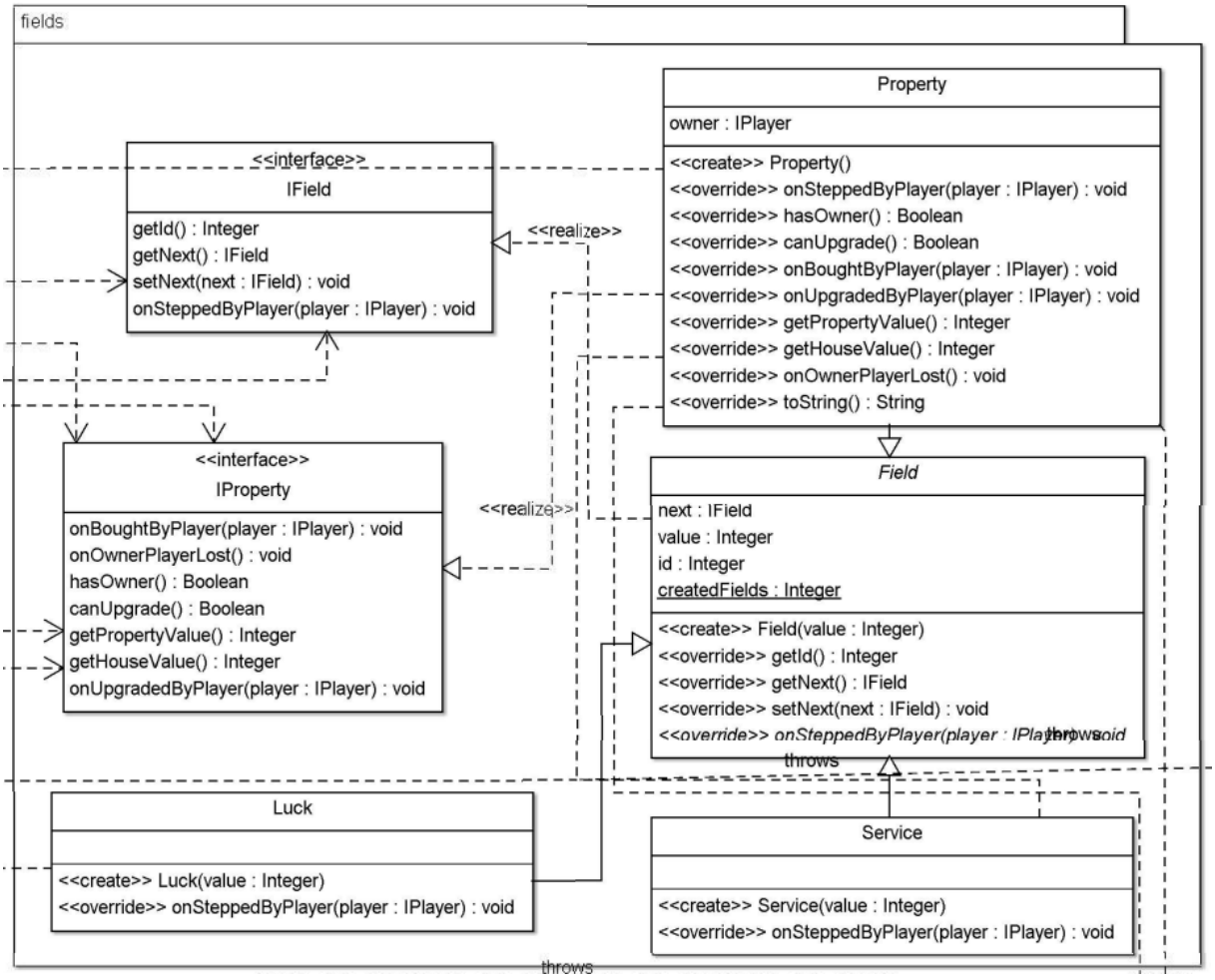
2.2.4 Greedy

A strategy metódust implementálja, amiben a „mohó” játékos logikája van.

2.2.5 Tactician:

A strategy metódust implementálja, amiben a „taktikus” játékos logikája van.

2.3 capitaly.fields package leírása:



2.3.1 IField:

Egy interfész, amit a csomagon kívűlről használnak az osztályok. A getNext metódus segítségével lehet végig iterálni a pályán. A getId metódus csak kiírás szempontjánól fontos. A setNext segítségével építí fel a pályát a Game osztály, illetve az onSteppedByPlayer metódus hívódik meg az adott mezőn, amikor rálép egy játékos. Ez a pálya egy mezőjét reprezentálja.

2.3.2 IProperty:

Ez az ingatlant reprezentáló interfész és ezt használja a játékos osztály, illetve a leszármazottjai.

2.3.3 Property:

Az ingatlant reprezentáló osztály, ami implementálja az IProperty interfészt és metódusait, illetve a Field osztályból származik és implementálja az absztrakt metódusait. Az osztály egy példányának egy birtokosa lehet egy játékos. Ezt az owner attribútum mutatja meg. Ha az értéke null, akkor még nincs birtokosa, egyébként van birtokosa.

2.3.4 Field:

Egy absztrakt típusa a mezőnek, ami implementálja az IField interfészt. Az onSteppedByPlayer metódust absztraktként beállítva a leszármazottaknak kell implementálniuk, hiszen ez a mező típusától függ, hogy mi történik ekkor. Az osztály a pálya egy mezőjét reprezentálja és egy láncolt listát képez a next attribútum segítségével. A Game osztály felel azért, hogy ez egy körpálya legyen. A value attribútum, pedig egy protected láthatóságú attribútum, ami a mező jelenlegi értékét reprezentálja.

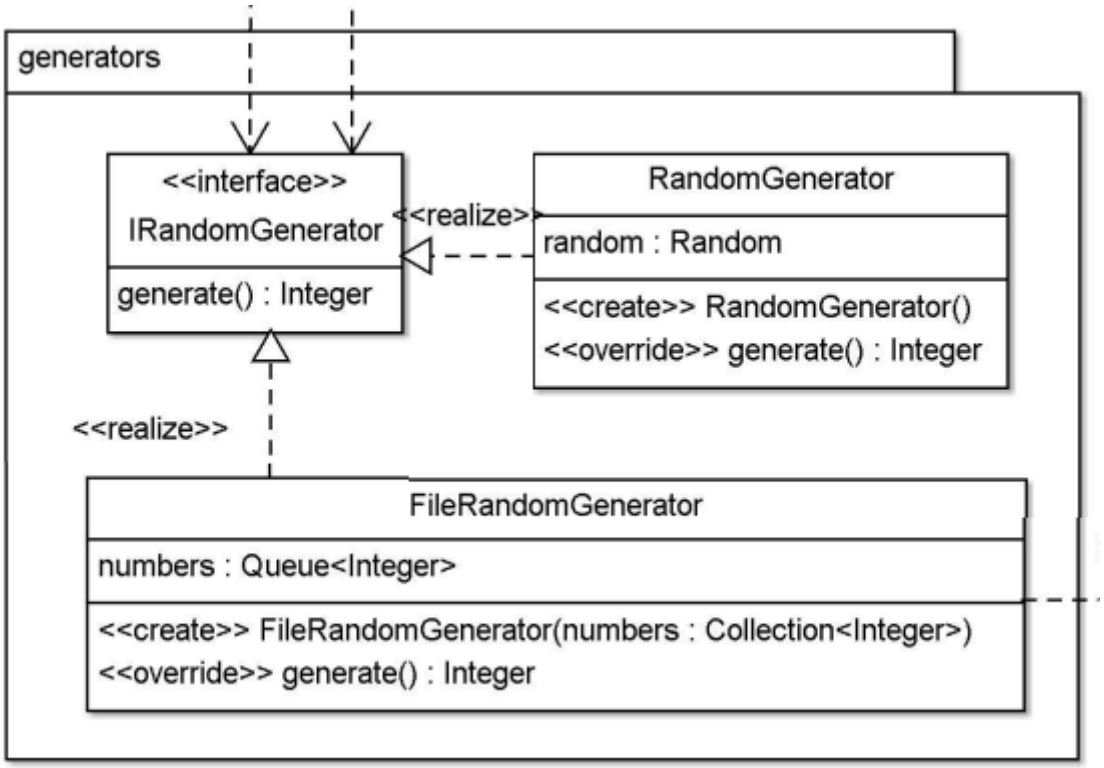
2.3.5 Luck:

A szerencse típusú mezőt reprezentálja. A Field osztályból származik és implementálja az onSteppedByPlayer metódusát, ahol a paraméterként megkapott játékoshoz hozzáadja a value attribútum értékét, amit egy konstruktorban kap meg.

2.3.6 Service:

A szolgáltatás típusú mezőt reprezentálja. A Field osztályból származik és implementálja az onSteppedByPlayer metódusát, ahol a paraméterként megkapott játékostól levonja a value attribútum értékét, amit egy konstruktorban kap meg.

2.4 capitaly.generators package leírása:



2.4.1 IRandomGenerator

Egy interfész, amit a csomagon kívűlről használnak az osztályok. Egy generate metódusa van, ami egy véletlenszerű számot generál.

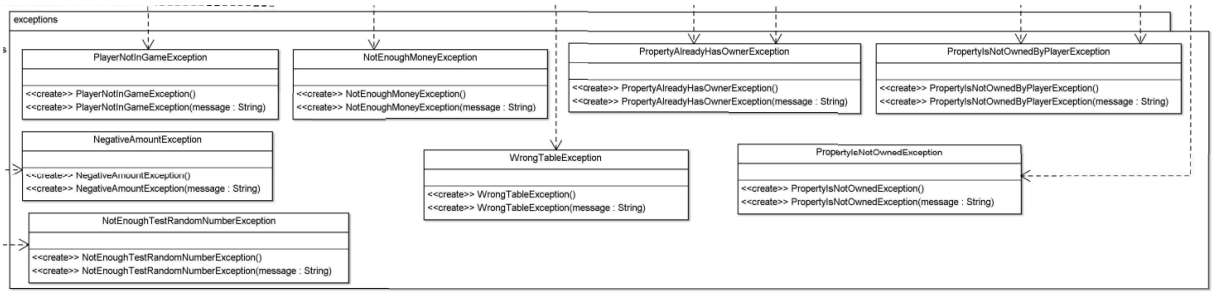
2.4.2 RandomGenerator:

Implementálja az IRandomGenerator interfészt. A generate metódusa ténylegesen egy véletlenszerű számot generál a Random típusú attribútuma segítségével.

2.4.3 FileRandomGenerator:

Implementálja az IRandomGenerator interfészt. A generate metódusa egy prioritásos sorból kiolvasott számot ad vissza, amit konstruktorban adnak át neki. Ez a tesztelhetőség miatt kell, hogy előre meghatározott számokat tudjon tesztelésnél „generálni” a program.

2.5 capitaly.exceptions package leírása:



2.5.1 PlayerNotInGameException:

Az Exception osztályból származik le és a Player osztály metódusai dobják ezt a kivételt, akkor, ha a játékos már nincs a játékban és úgy hívnak meg rajta egy adott metódust.

2.5.2 NegativeAmountException:

Az Exception osztályból származik le és a Player osztály metódusai dobják ezt a kivételt, akkor, ha a játékos addMoney vagy removeMoney metódusát negatív számmal hívják meg.

2.5.3 NotEnoughMoneyException:

Az Exception osztályból származik le és a Player osztály metódusai dobják ezt a kivételt, akkor, ha a játékos buy vagy upgrade metódusát hívják meg, de a játékos money attribútuma kisebb az ingatlan áránál.

2.5.4 PropertyAlreadyHasOwnerException:

Az Exception osztályból származik le és a Player és a Property osztály metódusai dobják ezt a kivételt, akkor, ha a játékos buy metódusát vagy az ingatlan onBoughtByPlayer metódusát hívják meg, de az ingatlannal már van tulajdonosa.

2.5.5 PropertyIsNotOwnedByPlayerException:

Az Exception osztályból származik le és a Player és a Property osztály metódusai dobják ezt a kivételt, akkor, ha a játékos upgrade metódusát vagy az ingatlan onUpgradedByPlayer metódusát hívják meg, de az ingatlannak nem tulajdonosa az adott játékos.

2.5.6 PropertyIsNotOwnedException:

Az Exception osztályból származik le és a Property osztály metódusai dobják ezt a kivételt, akkor, ha az ingatlan onUpgradedByPlayer metódusát hívják meg, de az ingatlannal nincs tulajdonosa.

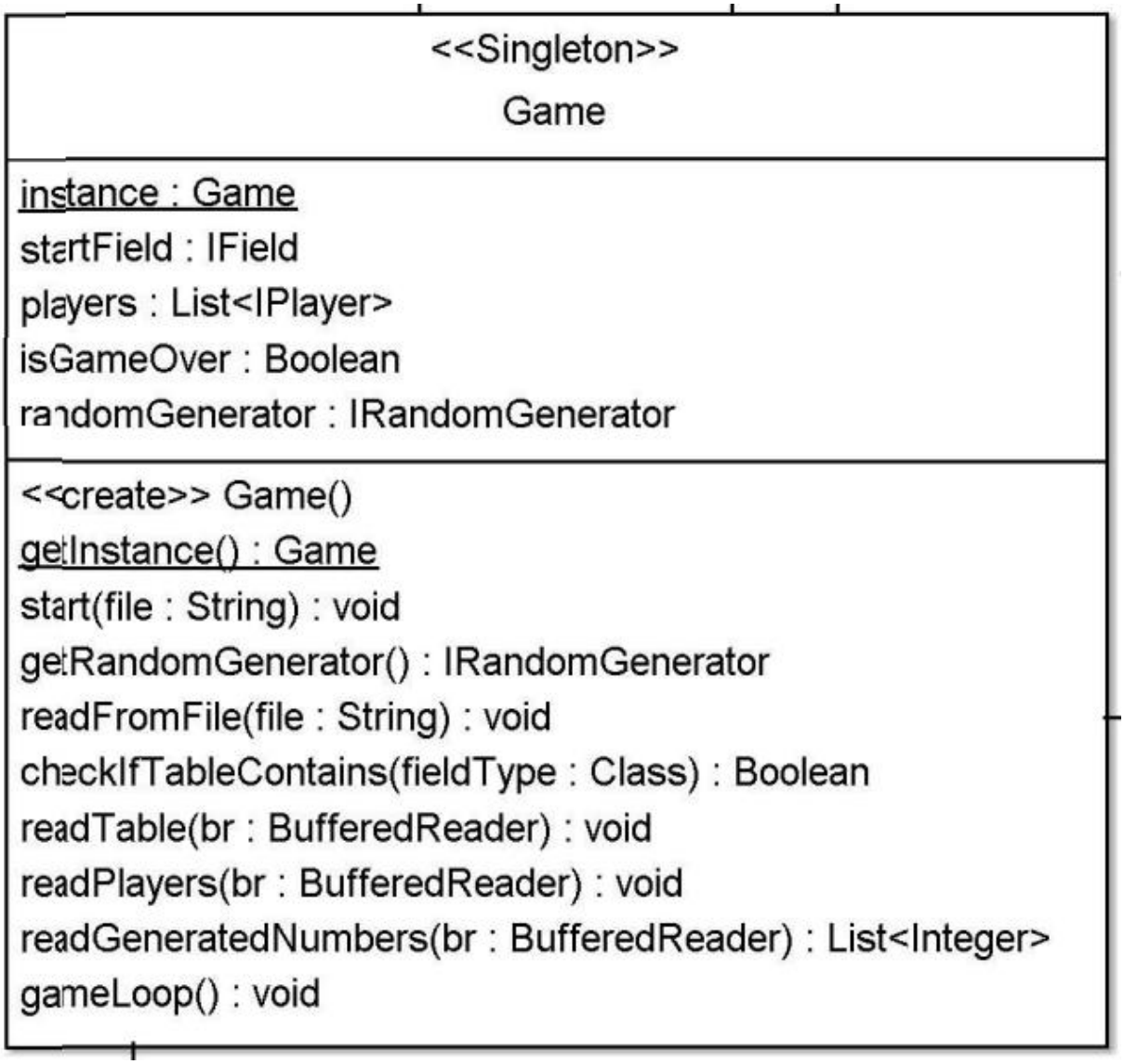
2.5.7 WrongTableException:

Az Exception osztályból származik le és a Game osztály dobja ezt a kivételt, amikor a pályát már beolvasta a játék, de nincsen legalább 1 mező a pályán az összes típusból.

2.5.8 NotEnoughTestRandomNumberException:

Az Exception osztályból származik le és a FileRandomGenerator osztály dobja ezt a kivételt, amikor nincs elég teszt szám, hogy véget érjen a játék, tehát, amikor a sorból kifogytak a számok.

2.6 capitaly package leírása:



2.6.1 Game:

A Game osztály egy Singleton osztály, tehát egy példány létezik az osztályból. Ez az osztály felel a játék inicializálásáért, illetve a játék levezényléséért.

Tesztelési terv:

Név	Leírás	Bemeneti fájl	Kimeneti fájl
Greedy_Test	Teszteli a „mohó” játékos teljes logikáját.	Input1.txt	Output1.txt
Tactician_Test	Teszteli a „taktikus” játékos teljes logikáját.	Input2.txt	Output2.txt
Cautious_Test	Teszteli a „óvatos” játékos teljes logikáját.	Input3.txt	Output3.txt
Service_Field_Test	Teszteli a „szolgáltatás” típusú mező logikáját.	Input4.txt	Output4.txt
Luck_Field_Test	Teszteli a „szerencse” típusú mező logikáját.	Input5.txt	Output5.txt
Property_Owned_ By_Player_without _house_and_other_ player_steps_on_it	Teszteli, hogy egy játékos által megvett ingatlanra (ház nélküli), ha rálép egy másik játékos, akkor tényleg 500 petákot von-e le a másik játékostól.	Input6.txt	Output6.txt
No_Input_File_Found	Teszteli, hogy mit ír ki a program, amennyiben nem található input fájl.	-	Output7.txt
Not_Enough_Random_ Test_Number_In_Input _File	Teszteli, hogy mit ír ki a program, ha nincs elegendő „kockadobás” a megadott input fájlban, tehát, ha nincsen vége a játéknak, mikor már elfogytak az input fájlban lévő „kockadobások”.	Input8.txt	Output8.txt
Invalid_Input_File_ Negative_Luck	Teszteli, hogy mit ír ki a program, amennyiben negatív számot adunk meg az egyik Luck típusú mezőnek.	Input9.txt	Output9.txt
Invalid_Input_File_ Negative_Service	Teszteli, hogy mit ír ki a program, amennyiben negatív számot adunk meg az egyik Service típusú mezőnek.	Input10.txt	Output10.txt
Property_Owned_ By_Player_without _house_and_owner_ player_steps_on_it	Teszteli, hogy egy játékos által megvett ingatlanra (ház nélküli), ha rálép a tulajdonos, akkor nem von-e le 500 petákot a játék.	Input11.txt	Output11.txt
Property_Owned_By_ Player_with_house_and_ other_player_steps_on_it	Teszteli, hogy egy játékos által megvett ingatlanra (házzal), ha rálép egy másik játékos, akkor tényleg 2000 petákot von-e le a másik játékostól.	Input12.txt	Output12.txt

Property_Owned_By_Player_with_house_and_owner_player_steps_on_it	Teszteli, hogy egy játékos által megvett ingatlanra (házzal), ha rálép a tulajdonos, akkor nem von-e le 2000 petákot a játék.	Input13.txt	Output13.txt
Invalid_Input_File_Not_All_Field_Types	Teszteli, hogy mit ír ki a program, amennyiben nincsen minden típusú mezőből legalább egy darab.	Input14.txt	Output14.txt