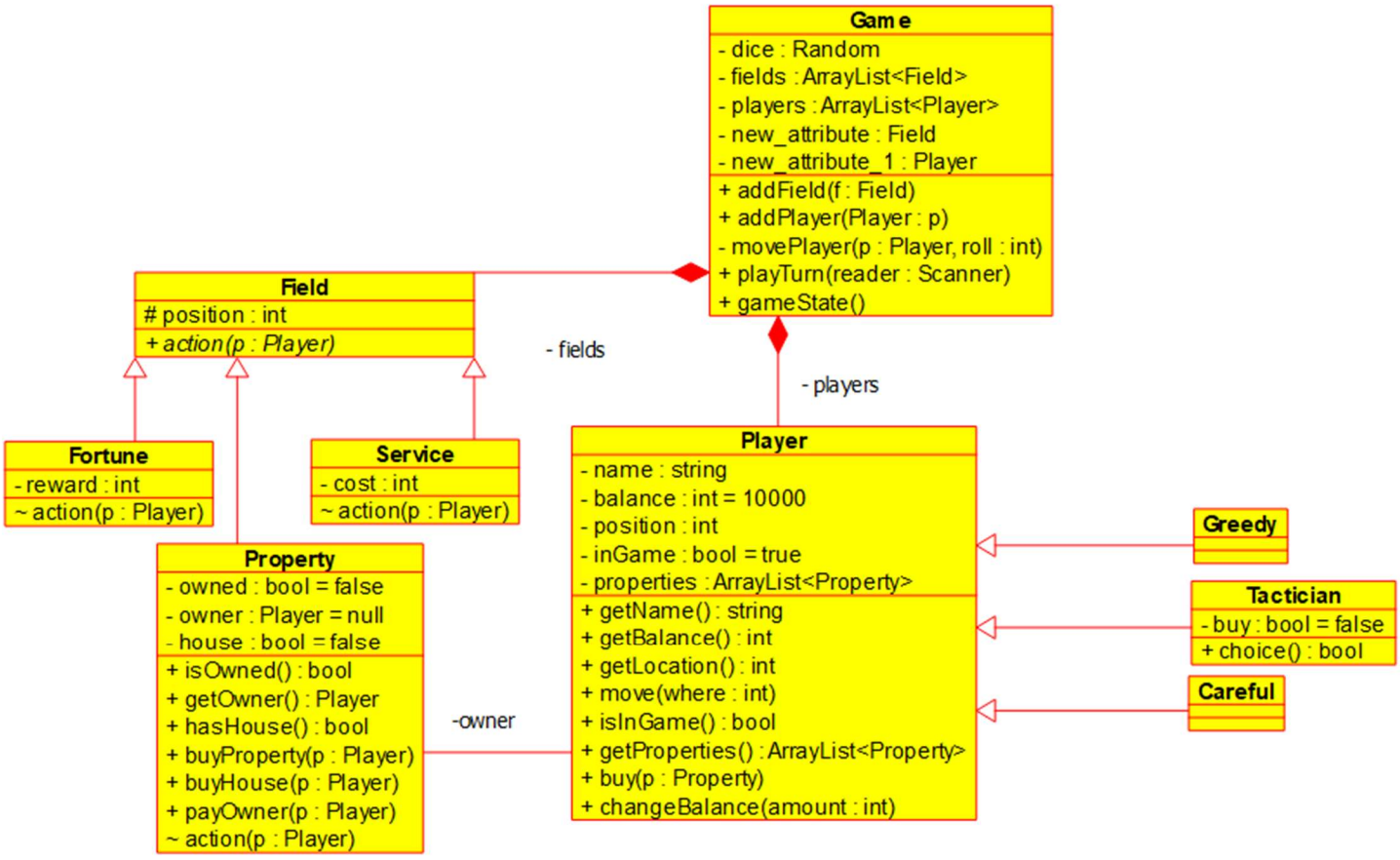
Progtech 1. beadandó dokumentáció

Készítette: Fiók Nándor (GSTQLI)

Feladat leírása:

3. feladat: 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!**

UML diagram:


Game
- dice : Random
- fields : ArrayList<Field>
- players : ArrayList<Player>
- new_attribute : Field
- new_attribute_1 : Player
+ addField(f : Field)
+ addPlayer(Player : p)
- movePlayer(p : Player, roll : int)
+ playTurn(reader : Scanner)
+ gameState()
Field
position : int
+ action(p : Player)
- fields
- players
Fortune
- reward : int
~ action(p : Player)
Service
- cost : int
~ action(p : Player)
Player
- name : string
- balance : int = 10000
- position : int
- inGame : bool = true
- properties : ArrayList<Property>
+ getName() : string
+ getBalance() : int
+ getLocation() : int
+ move(where : int)
+ isInGame() : bool
+ getProperties() : ArrayList<Property>
+ buy(p : Property)
+ changeBalance(amount : int)
Greedy
Tactician
- buy : bool = false
+ choice() : bool
Careful
Property
- owned : bool = false
- owner : Player = null
- house : bool = false
+ isOwned() : bool
+ getOwner() : Player
+ hasHouse() : bool
+ buyProperty(p : Player)
+ buyHouse(p : Player)
+ payOwner(p : Player)
~ action(p : Player)
-owner

Tesztesetek:

Bemenet:	Várt kimenet	Tényleges kimenet:
pálya mérete: 10 mezők listája: property property fortune 3000 service 2500 property fortune 1000 property service 1500 service 1000 property játékosok száma: 3 játékosok adatai: aladar greedy bela careful cecilia tactician körök száma: 10 dobások: 5 4 1 2 1 5 6 4 4 3 2 5 2 2 1 3 3 3 2 2 4 1 2 6 2 6 3 2 3 3	aladar: játékban, 11500 petákkal 5 és 7 mezőkön ház nélküli telkei vannak bela: játékban, 7000 petákkal 10-es mezőn ház nélküli telke van cecilia: játékban, 11000 petákkal 1 és 2 mezőkön ház nélküli telkei vannak	run: player standings after 10 rounds: ------------------------ name: aladar in game: yes balance: 11500 owned properties: position: 5 has house: no position: 7 has house: no ------------------------ name: bela in game: yes balance: 7000 owned properties: position: 10 has house: no ------------------------ name: cecilia in game: yes balance: 11000 owned properties: position: 1 has house: no position: 2 has house: no BUILD SUCCESSFUL (total time: 0 seconds)

pálya mérete: 7 mezők listája: fortune 1000 service 5000 property fortune 2000 service 3000 property fortune 3000 játékosok száma: 5 játékosok adatatai: aladar greedy adam greedy bela careful cecilia tactician csabi tactician körök száma: 12 6 4 5 4 2 5 5 5 6 2 1 4 2 1 2 4 5 6 1 1 4 1 2 6 6 3 2 6 5 2 3 1 4 5 1 2 1 6 1 5 3 5 6 5 5 6 3 5 5 6 5 4 4 6 5 2 4 2 5 1	aladar: kiesett adam: játékban, 12500 petákkal owned properties: 3. mezőn ház nélküli telke van bela: kiesett cecilia: játékban, 13000 petákkal 6 mezőn ház nélküli telke van csabi: játékban, 0 petákkal nincs egy telke sem	run: player standings after 12 rounds: ------------------------ name: aladar in game: no ------------------------ name: adam in game: yes balance: 12500 owned properties: position: 3 has house: no ------------------------ name: bela in game: no ------------------------ name: cecilia in game: yes balance: 13000 owned properties: position: 6 has house: no ------------------------ name: csabi in game: yes balance: 0 owned properties: none BUILD SUCCESSFUL (total time: 0 seconds)

pálya mérete: 15 mezők listája: property property fortune 2000 property property service 1000 property service 1000 fortune 3000 property service 1000 fortune 2000 service 2000 property property játékosok száma: 4 játékosok adatai: bela careful bob careful cecilia tactician csabi tactician körök száma: 7 dobások: 4 3 3 4 1 3 1 6 1 2 5 5 2 5 5 2 6 6 4 1 3 5 6 3 2 4 2 4	bela: játékban, 6500 petákkal owned properties: 4 és 5 mezőkön ház nélküli telkei vannak bob: játékban, 8500 petákkal nincs egy telke sem cecilia: játékban, 18000 petákkal 14-es mezőn ház nélküli telke van csabi: játékban, 9000 petákkal 10 és 2 mezőkön ház nélküli telkei vannak	run: player standings after 7 rounds: ------------------------- name: bela in game: yes balance: 6500 owned properties: position: 4 has house: no position: 5 has house: no ------------------------- name: bob in game: yes balance: 8500 owned properties: none ------------------------- name: cecilia in game: yes balance: 18000 owned properties: position: 14 has house: no ------------------------- name: csabi in game: yes balance: 9000 owned properties: position: 10 has house: no position: 2 has house: no BUILD SUCCESSFUL (total time: 0 seconds)

JavaDoc kommentek:

Game osztályon belül:

```
/**
 * A társasjáték létfontosságú adatait és metódusait foglalja magába.
 * @author fiokn
 */
public class Game

 /**
  * A dobáshoz szükséges random számgenerátor.
  */
 private final Random            dice    = new Random();
 /**
  * A pálya mezőit tartalmazó lista.
  */
 private final ArrayList<Field>  fields  = new ArrayList<>();
 /**
  * A játékosokat tartalmazó lista.
  */
 private final ArrayList<Player> players = new ArrayList<>();
 /**
  * f mezőt a pályához adja.
  * @param f
  */
 public void addField(Field f)   { fields.add(f);  }
 /**
  * p játékost a játékosok listájához adja.
  * @param p
  */
 public void addPlayer(Player p) { players.add(p); }
```

```
/**
 * p játékost roll mennyiséggel mozgatja el a pályán, számolva a túllépésekkel.
 * @param p
 * @param roll
 */
private void movePlayer(Player p, int roll)
```

```
/**
 * Szimulál egy teljes kört, azaz minden játékban lévő játékos dob, majd lép és taktikája szerint játszik.
 * Az előre megadott dobásokat a reader olvasóból nyeri ki, feltéve, hogy azok léteznek.
 * @param reader
 */
public void playTurn(Scanner reader)
```

```
/**
 * A feladat megoldásához szükséges kiíró metódus, mely a játékosok adatait és telkeit felsorolja.
 */
public void gameState()
```

Field osztályon belül:

```
/**
 * A különböző mezőket koordináló szülőosztály
 * @author fiokn
 */
public abstract class Field
```

```
 /**
  * A táblán belüli saját pozícióját tárolja, a telkek listázásához szükséges.
  */
 protected int position;
```

```
/**
 * p játékost taktikája szerint kiválogatja
 * @param p
 */
public void action(Player p)
```

Fortune osztályon belül:

```
/**
 * A szerencse mező.
 * @author fiokn
 */
public class Fortune extends Field
```

```
 /**
  * A mezőre lépőnek járó jutalom.
  */
 private final int reward;
```

Service osztályon belül:

```
/**
 * A szolgáltatás mező.
 * @author fiokn
 */
public class Service extends Field
{
    /**
     * A mezőre lépő játékos bűntetése.
     */
    private final int cost;
```

Property osztályon belül:

```
/**
 * Telek mező.
 * @author fiokn
 */
public class Property extends Field
{
    /**
     * A telek birtokoltsági állapotát jelző logikai változó.
     */
    private boolean owned = false;
    /**
     * A telek tulajdonosa. Ha nem birtokolja senki, null értéket vesz fel.
     */
    private Player  owner = null;
    /**
     * Megadja, hogy a telekre van-e ház építve.
     */
    private boolean house = false;
```

```
/**
 * basic getter
 * @return A telek birtokoltságát adja meg (logikai változó)
 */
public boolean isOwned()  { return owned; }
/**
 * basic getter
 * @return A telek tulajdonosát adja vissza. Tulajdonos hiányában null értéket ad vissza.
 */
public Player  getOwner() { return owner; }
/**
 * basic getter
 * @return Megadja, hogy van-e a telekre ház építve.
 */
public boolean hasHouse() { return house; }
/**
 * p játékos megveszi a telket, azaz számlája terhelődik, és az általa birtokolt telkek állománya bővül.
 * @param p
 */
public void buyProperty(Player p)
```

```
/**
 * p játékos házat épít telkére, melynek hatására számlája terhelődik.
 * @param p
 */
public void buyHouse(Player p)
```

```
/**
 * p játékos a telek tulajdonosának a ház létezésétől függően díjat fizet, ha nincs elég pénze,
 * akkor csak annyit, amennyije maradt.
 * @param p
 */
public void payOwner(Player p)
```

```java
/**
 * p játékos a telek tulajdonosának a ház létezésétől függően díjat fizet, ha nincs elég pénze,
 * akkor csak annyit, amennyije maradt.
 * @param p
 */
public void payOwner(Player p)
```

```java
/**
 * Az óvatos játékos döntési sorozata a mezőre lépés során.
 * @param c
 */
@Override
protected void action(Careful c)
```

```java
/**
 * Az óvatos játékos döntési sorozata a mezőre lépés során.
 * @param c
 */
@Override
protected void action(Careful c)
```

```java
/**
 * A mező adatainak kiírása a feladat megoldásakor.
 * @return String formában a telek adatait felsorolja.
 */
@Override
public String toString()
```

Player osztályon belül:

```java
/**
 * A játékosok általános adatait, cselekvéseit tároló osztály.
 * @author fiokn
 */
public abstract class Player
{
    /**
     * A játékos neve.
     */
    private final String              name;
    /**
     * A játékos egyenlege.
     */
    private int                       balance  = 10000;
    /**
     * A játékos pozíciója a játéktáblán.
     */
    private int                       location = -1;
    /**
     * Játékban van-e még a játékos?
     */
    private boolean                   inGame   = true;
    /**
     * A játékos által birtokolt telkek listája.
     */
    private final ArrayList<Property> properties = new ArrayList<>();
```

```java
/**
 * basic getter
 * @return A játékos nevét adja vissza (String típus).
 */
public String               getName()       { return name;       }
/**
 * basic getter
 * @return A játékos egyenlegét adja vissza (egész érték).
 */
public int                  getBalance()    { return balance;    }
/**
 * basic getter
 * @return A játékos táblán lévő pozícióját adja vissza.
 */
public int                  getLocation()   { return location;   }
/**
 * basic setter
 * @param where A megadott helyre mozgatja a játékost.
 */
public void                 move(int where) { location = where;  }
/**
 * basic getter
 * @return Megadja, hogy a játékos játékban van-e?
 */
public boolean              isInGame()      { return inGame;     }
/**
 * basic getter
 * @return A játékos telkeinek listáját adja vissza.
 */
public ArrayList<Property> getProperties() { return properties; }

/**
 * p telket a játékos által birtokolt listájához adja hozzá.
 * @param p
 */
public void                 buy(Property p) { properties.add(p); }
/**
 * amount mennyiséggel változtatja a játékos egyenlegét.
 * @param amount
 */
public void changeBalance(int amount)
```

Greedy osztályon belül:

```
/**
 * A mohó játékos.
 * @author fiokn
 */
public class Greedy extends Player
```

Careful osztályon belül:

```
/**
 * Az óvatos játékos.
 * @author fiokn
 */
public class Careful extends Player
```

Tactician osztályon belül:

```
/**
 * A taktikus játékos.
 * @author fiokn
 */
public class Tactician extends Player
{
    /**
     * Az adott körben vásárlási szándékát tároló logikai érték.
     */
    private boolean buy = false;

/**
 * basic getter
 * @return Megváltoztatja a játékos vásárlási szándékát, majd visszaadja azt.
 */
public boolean choice()
```