Programozás I. 1. zh

SZTE Szoftverfejlesztés Tanszék

2024. tavasz

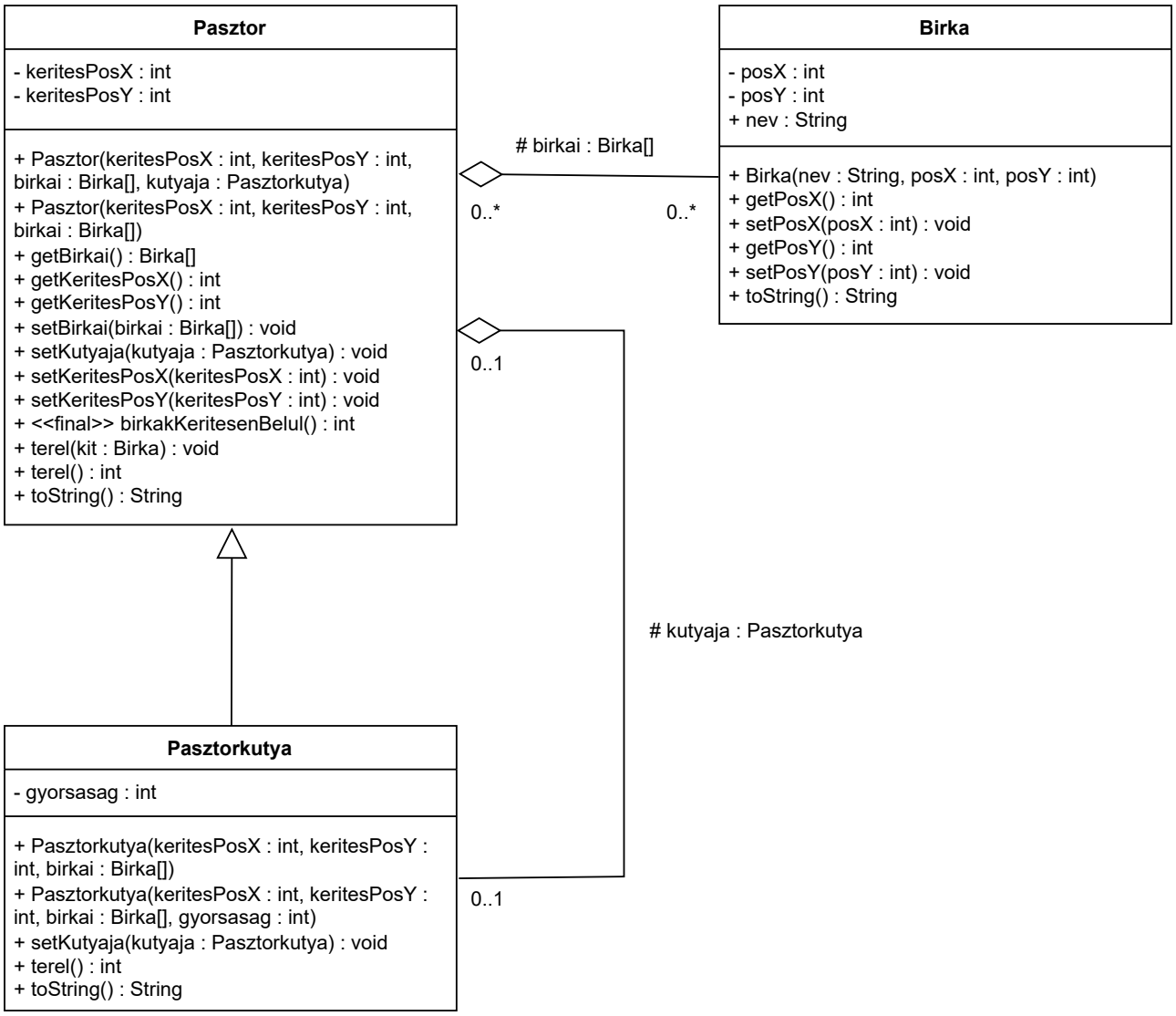
Általános követelmények, tudnivalók

- A feladat elkészítésére 30 perc áll a rendelkezésre. Ez szigorú határidő, a Bíró előre megadott időben zár.
- A feladatokat számítógép előtt kell megoldani, tetszőleges fejlesztői környezetben, tetszőleges operációs rendszer segítségével.
- Az elkészült programot **20** alkalommal lehet benyújtani, a megadott határidőig.
- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz pont.
  - Aki Windowst használ, annak a gép elindítása után érdemes egyből a fejlesztőkörnyezetet elindítani, és létrehozni egy új projektet, és csak utána a böngészőt, mivel az elején egy néhány percig indexel, addig pont el lehet olvasni a feladatot.
- Bármely segédanyag használata **tilos** (a fejlesztőkörnyezetek nyújtotta segítségen kívül), aki mégis ilyet tesz, vagy próbálkozik vele, annak a dolgozata nem értékelhető és a ZH nem teljesített. Ha valakinek a padtársa segít, akkor mérlegelés nélkül mindkettő hallgató dolgozata sikertelen, a ZH nem teljesített.
- A feladat megoldása során minden megadott előírást pontosan követni kell! Tehát, ha a feladat leírása szerint egy adattag neve a "elsoFoku", akkor az alábbi elnevezések nem megfelelőek: "elsőFokú", "elsofoku", "elso_foku", "elsőFoq". Ugyanez igaz a metódusok, osztályok elnevezésére is!
- A metódusok esetében a visszatérési típus, a név, módosítók és a paraméterek típusai (és azok sorrendje) kerülnek ellenőrzésre, azonban a paraméterek nevei tetszőlegesek lehetnek.
- A Java elnevezési konvenciókat követni kell (getter/setter elnevezés, toString, indentálás, stb). Abban az esetben is, ha ezt a feladat külön nem emeli ki, az ellenőrzés során erre is építünk.
- A nem forduló kódok nem kerülnek kiértékelésre, ezt utólagosan a gyakorlatvezető sem bírálhatja felül. (Hiszen mindenki rendelkezésére áll a saját környezete, ahol fordítani, futtatni tudja a forráskódot, így feltöltés előtt ezt mindenképpen érdemes megnézni!)
- Az adattagok és konstruktorok hiányában garantáltan 0 pontos lesz a kiértékelés, mert ezek minden teszt alapját képezik.
- Ha végtelen ciklus van a programban, akkor ezt a Bíró ki fogja dobni 3 másodperc után (ha többször is meghívásra kerül ilyen metódus, akkor ez többszöri 3 másodperc, összesen akár 2 perc is lehet). Ilyenkor NE kattints még egyszer a *Feltöltés* gombra, mert akkor

kifagyhat a Bíró, csak a böngésző újraindításával lehet megoldani a problémát (emellett elveszik 1 feltöltési lehetőség is).

- Kérdés/probléma esetén a gyakorlatvezetők tudnak segítséget nyújtani.
- **A feladat megoldása során a default csomagba dolgozz, majd a kész forrásfájlokat tömörítve, zip formátumban töltsd fel, azonban a zip fájlt tetszőlegesen elnevezheted!**
  - Zip készítése: Windowson és Linuxon is lehet a GUI-ban jobb klikkes módszerrel tömörített állományt létrehozni (Windowsban pl. a 7-Zip nevű ingyenes program használatával).
  - Linux terminálon belül például a "zip feladat.zip *.java" paranccsal is elkészíthető a megfelelő állomány.
- A feladatokban az alábbi dolgok az alapértelmezettek (**kivéve**, ha a feladat szövege mást mond)
  - az osztályok láthatósága publikus
  - az egész érték 32 bites
  - a lebegőpontos számok dupla pontosságúak
  - az olyan metódusok void visszatéréssel rendelkeznek, amelyeknél nincs specifikálva visszatérési típus.
  - a metódusok mindenki számára láthatóak
  - az adattagok csak az adott osztályban legyenek elérhetőek
- A *riport.txt* és a fordítási log fájlok megtekinthetőek az alábbi módon:
  1. Az *Eredmények megtekintése* felületen a vizsgálandó próba új lapon való megnyitása
  2. A kapott url formátuma: `https://biro.inf.u-szeged.hu/Hallg/IB204L-1/1/hXXXXXX/4/riport.txt`
  3. Az url-ből visszatörölve a 4-esig (`riport.txt` törlése) megkaphatók a 4-es próbálkozás adatai.
- Szövegek összehasonlításánál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")
- A leírásokban bemutatott példákban a stringek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül a `3 alma`, de a szóköz szükséges!
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

1. ábra. Osztálydiagram



Birka osztály (4 pont)

Az első osztály a `Birka`, amely egy terelhető birkát reprezentál. Minden birka rendelkezik egy csak az osztályon belül látható x (`posX`) és y (`posY`) koordinátával, itt van éppen a birka. Ezen kívül minden birkának van egy publikus láthatóságú szöveges `nev` adattagja is.

A konstruktor és a szükséges metódusok:

- Konstruktor, amely a három tulajdonságát várja (és állítja be) a következő sorrendben: nev, posX, posY.
- `getPosX()` metódus, ami visszaadja az posX értékét.
- `getPosY()` metódus, ami visszaadja az posY értékét.

- `setPosX(int posX)` metódus, beállítja a posX értékét.
- `setPosY(int posY)` metódus, beállítja a posY értékét.
- `toString()` metódus, ami a következő formába alakítja a birkát: "<nev>[<posX>,<posY>]" (Például "Shaun[2,0]").

Pasztor osztály (11 pont)

Készítsd el a `Pasztor` osztályt, amely egy pásztort reprezentál. Minden pásztor rendelkezik kerítés koordinátákkal, ezek a privát láthatóságú `keritesPosX` és `keritesPosY` egész szám változók. A pásztornak van továbbá egy saját maga és leszármazottai által látható `birkak` tömbje, amely birkákat tárol, a teljes programban feltételezhetjük, hogy ez mindig tele van, nem tartalmaz `null` értékeket. Egy szintén védett láthatóságú Pasztorkutya típusú `kutyaja` objektuma is van, ez viszont lehet majd `null`, ha nincs kutyája.

A konstruktor és a szükséges metódusok:

- Konstruktor, amely a pásztor mind a négy tulajdonságát várja (és állítja be) a következő sorrendben: keritesPosX, keritesPosY, birkai, kutyaja. A kutyaja beállításánál érdemes a settert használni.
- Konstruktor, amely a pásztor három tulajdonságát várja (és állítja be) a következő sorrendben: keritesPosX, keritesPosY, birkai. A kutyaja értéket `null`-ra állítja.
- `getBirkai()` metódus, amely visszaadja a birkai tömböt.
- `getKeritesPosX()` metódus, amely visszaadja a keritesPosX értéket.
- `getKeritesPosY()` metódus, amely visszaadja a keritesPosY értéket.
- `setBirkai(Birka[] birkai)` metódus, amely beállítja a birkai tömböt.
- `setKutyaja(Kutya kutyaja)` metódus, amely beállítja a kutyaja objektumot. A kutya szintén egy pásztornak minősül, ezért érdemes lehet ezen a ponton a Pasztorkutya osztályt (és getter-setter metódusait) megvalósítani (lásd lentebb). Az öröklődés miatt a Pasztorkutya örökli a Pasztor összes tulajdonságát. A setter mielőtt beállítja a kutyát az aktuális pásztornak, állítsa át a kutya kerítés pozícióit és birkáit az aktuális pásztor ugyanilyen adattagjaira, a kutya settereinek felhasználásával.
- `setKeritesPosX(int keritesPosX)` metódus, amely beállítja a keritesPosX értéket. Ha a kutya értéke nem `null`, akkor a kutya ugyanilyen értékét is állítsa át annak setterével.
- `setKeritesPosY(int keritesPosY)` metódus, amely beállítja a keritesPosY értéket. Ha a kutya értéke nem `null`, akkor a kutya ugyanilyen értékét is állítsa át annak setterével.

- `toString()` metódus, amely szöveggé alakítja a pásztort a következő formában: "A kerítés koordinátái <keritesPosX>,<keritesPosY>, birkái: [<birkai>]", ahol a megfelelő adattagokat helyettesítjük be. A birkai felsorolásnál a tömb összes elemének toString-jét hívjuk meg, ezt legkönnyebben az `Arrays.toString(birkai)` beépített Java metódussal tudjuk megtenni. (Például: "A kerítés koordinátái 1,1, birkái: [Kolmogorov[1,1], Origo[0,0], Shaun[3,3], Commander Shepard[0,1]]").

- `birkakKeritesenBelul()` metódus, azt adja vissza, hogy hány darab birka van a birkak tömbben, amely x és y pozíciója megegyezik a jelenlegi pásztor kerítésének x és y pozíciójával. A metódus nem felüldefiniálható a leszármazott osztályokban.

- `terel(Birka kit)` metódus, amely a paraméterben kapott birkát a kerítés felé tereli. Ennek során a metódus lekéri a birka aktuális x és y pozícióját, és mindkettőt megváltoztatja (ha kell), hogy 1 pozícióval közelebb kerüljön a kerítéshez. (Például keritesPosX: 1, keritesPosY: 1 kerítés és a birka posX: 4, posY: -1 esetben a birka új pozíciója posX: 3, posY: 0).

- `terel()` metódus, amely felhasználja az előző két metódust: Először lementi, hogy kezdetben hány birka van jó helyen a birkakKeritesenBelul() segítségével. Ezután végigmegy a pásztor birkak tömbjén, és a **legelső** olyan birkát (ha van ilyen), amely nem a kerítés pozícióján van éppen, tereli a terel(Birka kit) metódus segítségével. Ha a pásztor egy birkát már terelt, akkor ebben a metódusban nem fog többet terelni. Ha van kutyája (azaz az objektum nem `null`), akkor a kutya `terel()` metódusát is hívjuk meg ezután! A terelés metódusunk végeztével számoljuk ki és adjuk vissza azt, hogy a teljes terelés során hány új birka került a kerítés pozíciójára (a korábbi és új birkakKeritesenBelul() érték különbségéből könnyen megkapható).

  A metódus működésére a **??**. és a **??**. ábrán láthatunk példákat, az előbbin egy pásztor kutya nélkül, míg az utóbbin egy pásztor kutyával látható. Itt minden sor egy újabb terelés, a világosszöld háttér jelzi, hogy jó helyen van a birka, a rombusz pedig, hogy történt változás a terelés során az adott birkánál. A nyíl a kimenetet jelzi. A **??**. ábrán a kutya terelésének kimenetét nem írtuk fel, habár ez a kutya lépés utáninak néz ki, a teljes pásztor terelést mutatja (a pásztorét, és a kutyájáét egymás után), azaz a két sor összes helyre került birkáját.

Pasztorkutya osztály (5 pont)

Készítsd el a `Pasztorkutya` osztályt, amely a `Pasztor` osztályból származik. Rendelkezik egy gyorsasag privát láthatóságú egész szám tulajdonsággal, amelyekhez nem szükséges getter és setter metódus.

A konstruktor és a szükséges metódusok:

- Konstruktor, amely megkapja (és beállítja) a kutya x és y kerítés pozícióját és birkáit, ebben a sorrendben: keritesPosX, keritesPosY, birkai. Ehhez felhasználja a szülőjének konstruktorát is. A gyorsaság értéke alapból 1 legyen, a kutyaja értéke pedig `null`.

- Konstruktor, amely megkapja (és beállítja) a kutya x és y kerítés pozícióját, birkáit, és gyorsaságát ebben a sorrendben: keritesPosX, keritesPosY, birkai, gyorsasag. Ehhez felhasználja a szülőjének konstruktorát is. A kutyaja értéke legyen `null`.

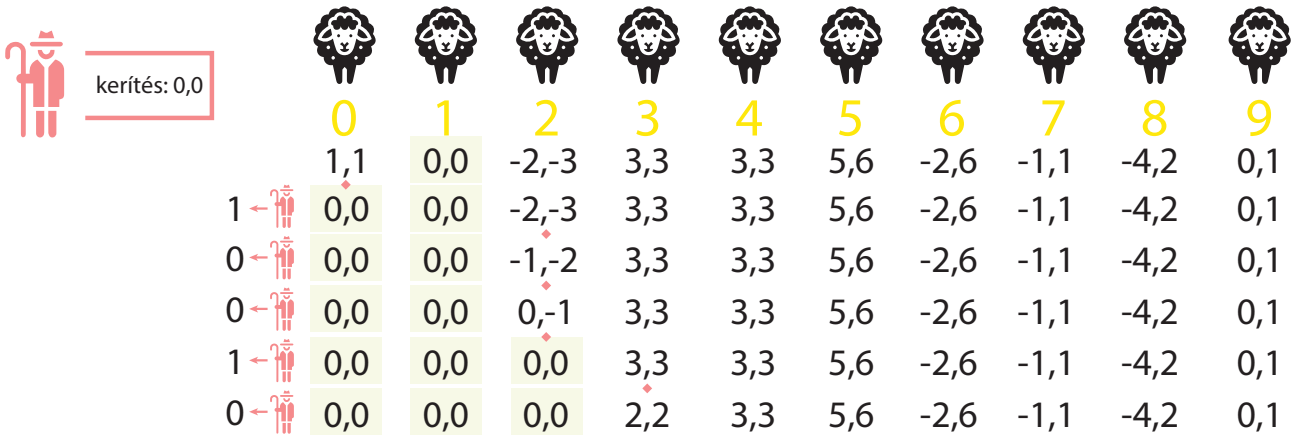
- Definiáljuk felül a `setKutyaja(Pasztorkutya kutyaja)` metódust, hogy ne a kapott kutyára, hanem `null`-ra állítsa az aktuális pásztorkutya kutyaja tulajdonságát, mivel a kutyának nem lehet kutyája. A paraméterben kapott kutyával nem csinál semmit a metódus.

- Definiáljuk felül a `toString()` metódust, hogy hívja meg a szülőjének az ugyanilyen metódusát, de a következő szöveggel egészítse ki annak eredményét: ". Ő egy <gyorsasag> gyorsaságú pásztorkutya.". (Például a teljes visszaadott érték: "A kerítés koordinátái 1,1, birkái: [Kolmogorov[1,1], Origo[0,0], Shaun[3,3], Commander Shepard[0,1]]. Ő egy 2 gyorsaságú pásztorkutya.").

- Definiáljuk felül a `terel()` metódust, hogy gyorsasag számú terelést hajtson végre. Ennek során szintén mentsük le a most kerítés pozición lévő birkákat (felhasználva a birkakKeritesenBelul() metódust), majd az összes birkán végighaladva a birkákon az első <gyorsaság> darab birkára hívjuk meg a terelést. (Például 2 gyorsaság esetén az első két olyan birkát tereljük egy-egy alkalommal, amely nem jó pozición van.). Ezután itt nincs szükség másra, mint kiszámolni az újonnan a kerítés pozíciójára került birkákat, ebben szintén a birkakKeritesenBelul() metódust használhatjuk fel. Ezt a különbséget adjuk vissza!

  A metódus működésére a **??**. és a **??**. ábrán láthatunk példákat, az előbbin egy kutya egymaga, míg az utóbbin egy pásztor kutyával látható.

2. ábra. Példa terelésre egy [0,0] kerítésű pásztor esetében.



Jó munkát!

3. ábra. Példa terelésre egy 2 gyorsaságú, [0,0] kerítésű pásztorkutya esetében.

	0	1	2	3	4	5	6	7	8	9
	1,1	0,0	-2,-3	3,3	3,3	5,6	-2,6	-1,1	-4,2	0,1
1 ←	0,0	0,0	-1,-2	3,3	3,3	5,6	-2,6	-1,1	-4,2	0,1
0 ←	0,0	0,0	0,-1	2,2	3,3	5,6	-2,6	-1,1	-4,2	0,1
1 ←	0,0	0,0	0,0	1,1	2,2	5,6	-2,6	-1,1	-4,2	0,1
1 ←	0,0	0,0	0,0	0,0	1,1	5,6	-2,6	-1,1	-4,2	0,1
1 ←	0,0	0,0	0,0	0,0	0,0	4,5	-2,6	-1,1	-4,2	0,1

4. ábra. Példa terelésre egy [1,1] kerítésű pásztor és 2 gyorsaságú kutyája kutyája esetében.

	0	1	2	3	4	5	6	7	8	9
	1,1	0,0	-2,-3	3,3	3,3	5,6	-2,6	-1,1	-4,2	0,1
	1,1	1,1	-2,-3	3,3	3,3	5,6	-2,6	-1,1	-4,2	0,1
1 ←	1,1	1,1	-1,-2	2,2	3,3	5,6	-2,6	-1,1	-4,2	0,1
	1,1	1,1	0,-1	2,2	3,3	5,6	-2,6	-1,1	-4,2	0,1
1 ←	1,1	1,1	1,0	1,1	3,3	5,6	-2,6	-1,1	-4,2	0,1
	1,1	1,1	1,1	1,1	3,3	5,6	-2,6	-1,1	-4,2	0,1
1 ←	1,1	1,1	1,1	1,1	2,2	4,5	-2,6	-1,1	-4,2	0,1
	1,1	1,1	1,1	1,1	1,1	4,5	-2,6	-1,1	-4,2	0,1
1 ←	1,1	1,1	1,1	1,1	1,1	3,4	-1,5	-1,1	-4,2	0,1
	1,1	1,1	1,1	1,1	1,1	2,3	-1,5	-1,1	-4,2	0,1
0 ←	1,1	1,1	1,1	1,1	1,1	1,2	0,4	-1,1	-4,2	0,1
	1,1	1,1	1,1	1,1	1,1	1,1	0,4	-1,1	-4,2	0,1
1 ←	1,1	1,1	1,1	1,1	1,1	1,1	1,3	0,1	-4,2	0,1
	1,1	1,1	1,1	1,1	1,1	1,1	1,2	0,1	-4,2	0,1
2 ←	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	-4,2	0,1
	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	-3,1	0,1
1 ←	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	-2,1	1,1
	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	-1,1	1,1
0 ←	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	0,1	1,1
	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1
1 ←	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1