Programozás I. Gyakorló feladatsor

SZTE Szoftverfejlesztés Tanszék

2024. tavasz

Általános követelmények, tudnivalók

- A feladat elkészítési határideje: **vasárnap 23:59:59**. Ez szigorú határidő, a Bíró előre megadott időben zár, pótlásra nincs lehetőség.
- 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.
- 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://biro2.inf.u-szeged.hu/Hallg/IB204L/FELADAT/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).

Aréna

Előfeltételek: A feladat megoldásához szükségesek a hibátlanul működő Sarkany, Ember és Hos osztályok.

1. feladat (13 pont)

Készítsd el a `Arena` osztályt! Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságok használatára!

A beállított sárkányt utólag ne lehessen módosítani.

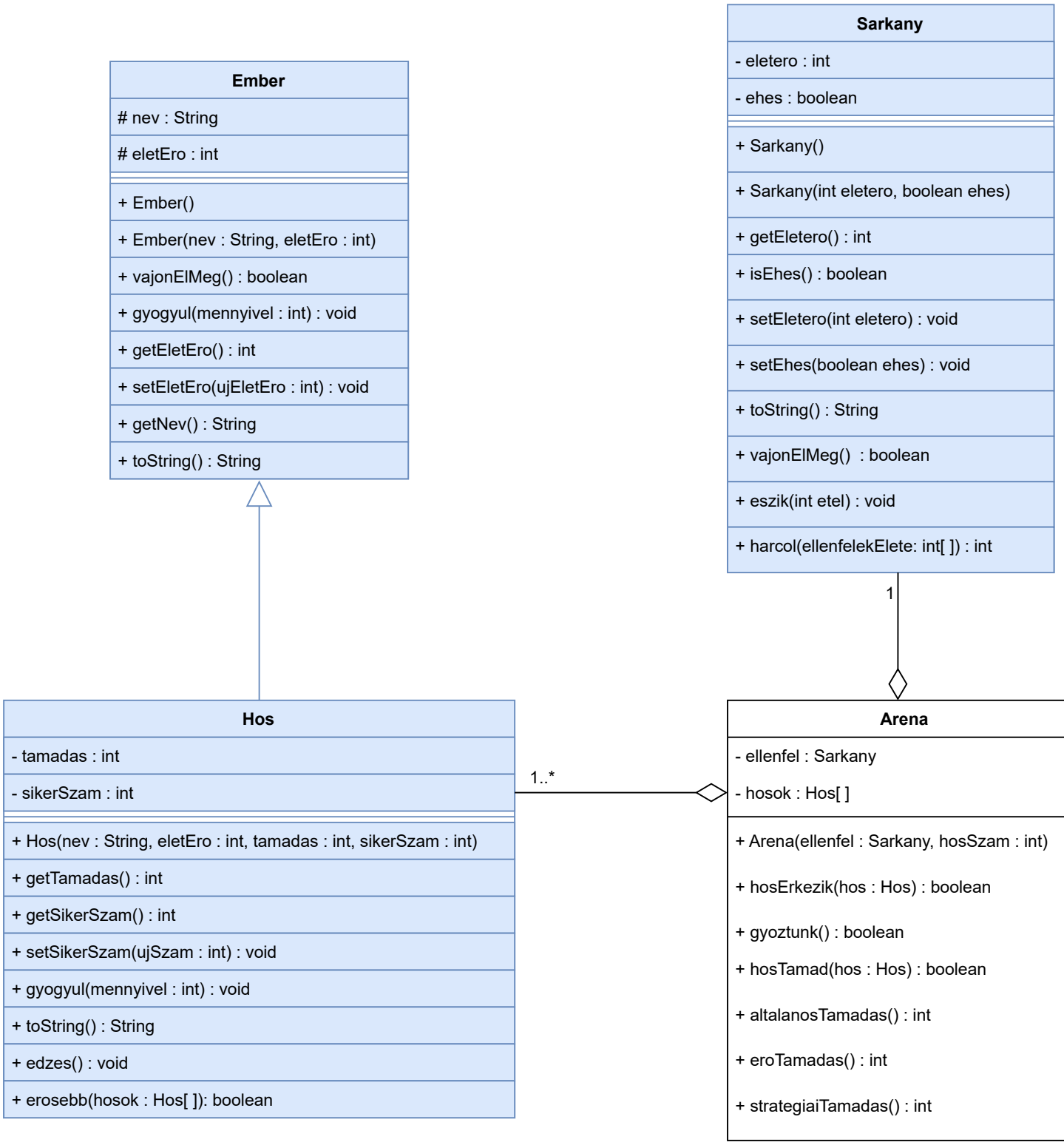
A *paraméteres konstruktor* hozza létre a *hosok* tömböt, melynek maximális mérete *hosSzam*. **(2 pont)**
Emlékeztető: a hősök számának pozitívnak kell lennie. Amennyiben valaki nem ilyen értéket ad meg, akkor a *hosok* tomb mérete legyen 1.

Készítsd el a *hosErkezik()* metódust!

- amennyiben a tömbben még van olyan pozíció, ami null, akkor az első ilyen helyre tedd be a paraméterben kapott hőst, és térj vissza igazzal
- amennyiben a tömbben már nincs olyan pozíció, ami null, akkor írd ki a hibacsatornára, hogy "az arena megtelt", majd térj vissza hamissal. A kiírást zárja sortörés! **(3 pont)**

Készítsd el a *gyoztunk()* metódust! A metódus hívja meg az ellenfél sárkány *VajonElMeg()* metódusát, és ha él, akkor adjon vissza hamisat, ha nem él, akkor térjen vissza igazzal. **(1 pont)**

1. ábra. UML Osztálydiagram



Készítsd el a *hosTamad()* metódust!

- ha a hős halott, ne csinálj semmit, csak adjon vissza hamis értéket.

- ha a hős él, de az ellenfél sárkány halott, akkor adjon vissza igaz értéket.
- ha mindketten élnek, akkor a sárkány életerejéből vond le a hős támadását. Ha ekkor a sárkány meghalt, adj vissza igaz értéket, és növeld eggyel a hős sikerSzámát. Ha a sárkány ekkor még mindig él, akkor hívd meg az *eszik()* metódust a hős életerejével, és nullázd le a hős életerejét, majd adj vissza hamis értéket. **(1 pont)**

Készítsd el az *altalanosTamadas()* metódust!
A metódus menjen végig a *hosok* tömbön, egészen addig, míg a sárkány meghal, vagy a tömb véget nem ér, és minden hősre hívja meg a *hosTamad()* metódust. A végén azt a számot adja vissza, hogy hány hős halt meg a támadás során. **(2 pont)**

Megjegyzés: ez a metódus a halott hősöket is harcba küldi, és őket is beleszámolja az áldozatok közé.

Készítsd el az *eroTamadas()* metódust!
A metódus válassza ki a legnagyobb támadású, még élő hőst, és azzal hívja meg a *hosTamad()* metódust. Ha a sárkány még él, akkor hívja meg a következő legnagyobb támadású, még élő hőst, és így tovább, meg a sárkány meg nem hal, vagy az összes hős el nem esik. A végén azt a számot adja vissza, hogy hány hős halt meg a támadás során. **(2 pont)**

Készítsd el a *strategiaiTamadas()* metódust!
A metódus menjen végig a *hosok* tömbön, egészen addig, míg a sárkány meghal, vagy a tömb véget nem ér, és amennyiben a hős él, illetve a támadása nagyobbb, mint az életereje, akkor a hősre hívja meg a *hosTamad()* metódust. A végén azt a számot adja vissza, hogy hány hős halt meg a támadás során. **(2 pont)**

Tipp: ügyelj arra, hogy a *hosok* tömbben előfordulhatnak null értékek. Ezeknek nyilván ne hívd meg a kért metódusát, mert az futási hibát fog eredményezni, amire csak részpont adható.

Megjegyzés: az *altalanosTamadas()*, *eroTamadas()* és *strategiaiTamadas()* metódusok tesztjeinek futtatásához szükséges a *hosErkezik()* metódus megléte. Annak működése a fent említett tesztek futtatásához lényegtelen, akár egy konstans true / false érték visszaadásával működik.

Jó munkát!