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")

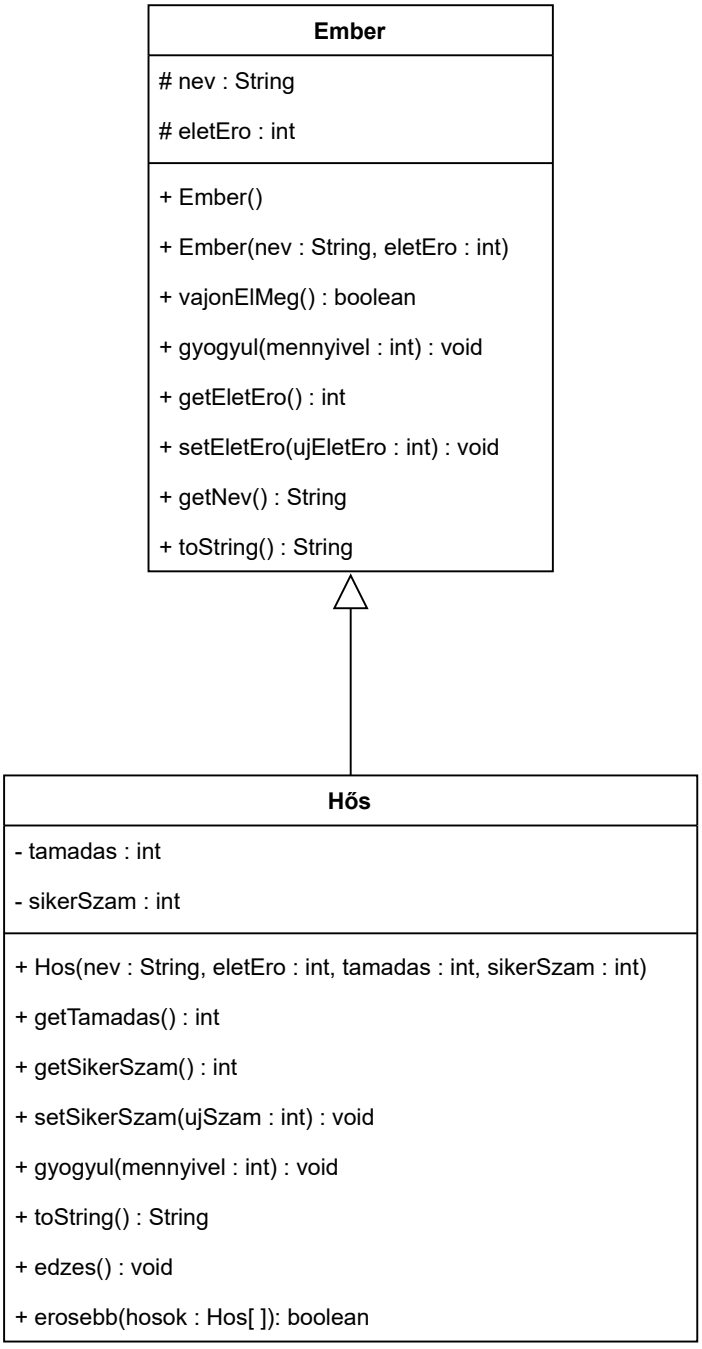
- A leírásokban bemutatott példákban a stringek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül a `3 alma`, de a szóköz szükséges!

- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

Ember (10 pont)

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

1. ábra. UML Osztálydiagram



Az emberek életereje minden esetben csak nemnegatív egész értékek lehetnek.
Az `setEletero` paraméterben egy egész számot vár, ha negatív számot kap, akkor az életerőt 0-ra állítja, egyéb esetben pedig a paraméterben érkező értékre. **(1 pont)**

A getter metódusok értelemszerűen működjenek. **(1 pont)**

A *default konstruktor* "ismeretlen" nevű, 10 életerővel rendelkező embert inicializáljon. **(2 pont)**

A paraméteres konstruktor működjön értelemszerűen. Ügyelj rá, hogy az életerő nemnegatív. Ha negatív számot kap, akkor az ellentettjét állítsa be, azaz -8 helyett legyen 8. **(2 pont)**

A *toString()* metódus az ember adatait adja vissza, az alábbi formában:
"*Emberunk neve {nev}, es jelenleg {allapot}.*"
A *{nev}* helyére az ember neve kerüljön, az *{allapot}* helyére pedig egy kifejezés az alábbi módon:

- ha 10-nél nagyobb az életereje, akkor "majd kicsattan az egeszsegtol"
- ha az életereje 1 és 10 között van, akkor "atlagos az allapota"
- ha az életereje 0, akkor "halott"

(2 pont)

A *vajonElMeg()* metódus térjen vissza igazzal, ha az ember életereje nagyobb, mint 0. Egyéb esetben hamissal térjen vissza. **(1 pont)**

A *gyogyul()* metódus az alábbiak szerint működjön: ha az életerő 0, akkor a metódus írja ki a hibacsatornára a "Sajnalom, elkestetek." szöveget. A kiírást zárja sortörés. Minden más esetben a metódus növelje meg az életerőt a paraméterben kapott értékkel. **(1 pont)**

Hős (14 pont)

Készítsd el a `Hos` osztályt. **(1 pont)**
A hősök adattagjai nemnegatív egész értékek, erre ügyelj minden beállításkor.
Az adattagok jelentése:

- *tamadas*: a hős támadóereje
- *sikerSzam*: az eddig megölt sárkányok száma

Az adattagokhoz legyenek lekérő és beállító metódusok. **(1 pont)**

A *sikerSzam* működése: amennyiben az új érték kevesebb, mint a sikerSzam jelenlegi értéke, akkor ne csináljon semmit, ellenkező esetben állítsa be az új értéket. **(1 pont)**

Készíts *paraméteres konstruktor*t.**(3 pont)**

Emlékeztető: a támadás és a sikeresen megölt sárkányok száma is nemnegatív. Ha negatív érték érkezik, állítsuk 0-ra.

A *toString()* működése: adja vissza mind az ős, mind a gyerek adatait, az alábbi formában:
"*Emberunk neve {nev}, es jelenleg {allapot}. Ez az ember egy sarkanyolo hos, tamadasa {tamadas}, es eddig {sikerSzam} darab sarkanyt olt meg.*"
Használd fel az ősosztályban definiált `toString` metódust. **(2 pont)**

A *gyogyul()* metódus működése: a hősök mindenképpen gyógyulnak, még akkor is, ha nulla az életerejük.**(2 pont)**

Az *edzes()* metódus működése: ha az adott hős még él, akkor növelje meg 1-gyel a támadást. Ellenkező esetben ne csináljon semmit.**(2 pont)**

Az *erosebb()* metódus működése: számítsa ki, hogy az adott hős erősebb-e a paraméterben lévő összes hősöknél. Egy hős akkor erősebb egy másiknál, ha nagyobb a támadóereje, vagy megegyező támadóerő esetén nagyobb az életereje. **(2 pont)**
Jó munkát!