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).

STORY

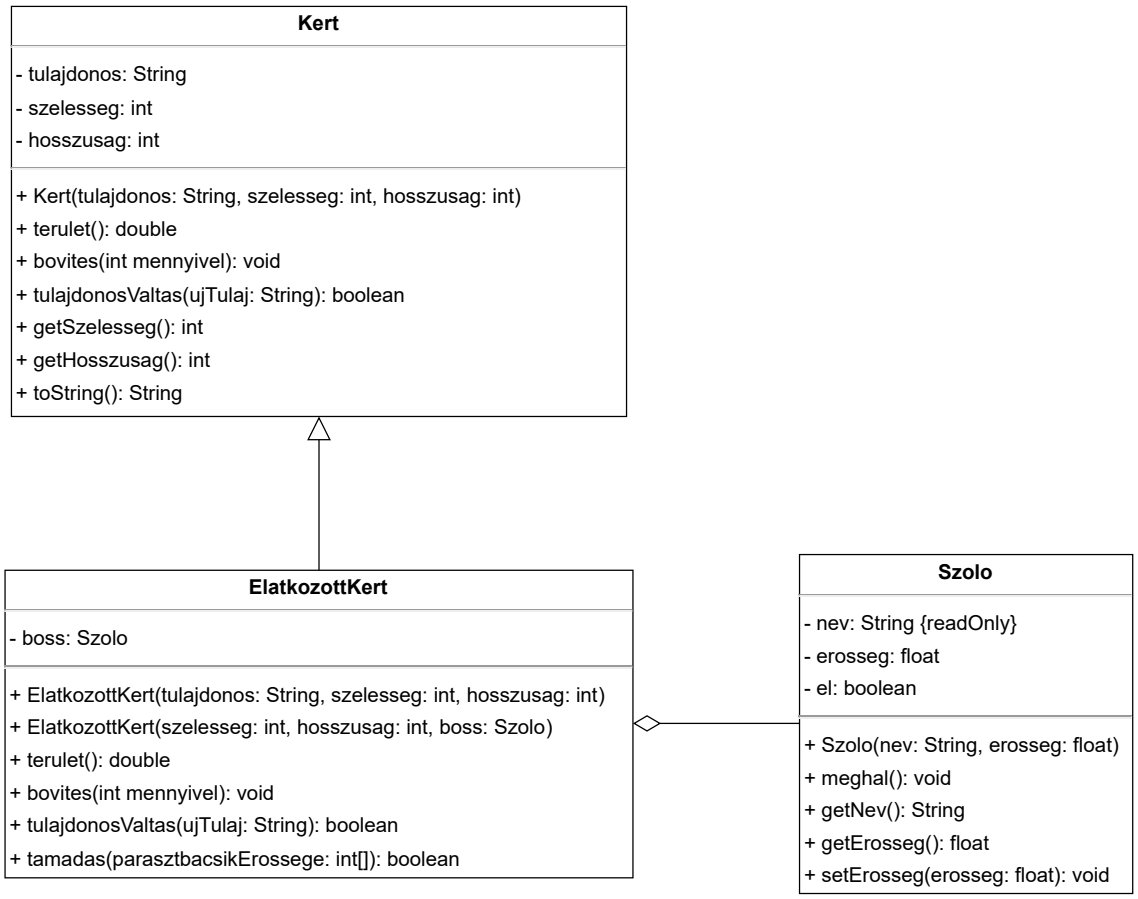
Az átok

Grapeliena egy boszorkány, akinek célja, hogy a világon ne legyen más gyümölcs, csak szőlő, ugyanis a szőlők hatalmas varázserővel rendelkeznek, amelyet kinyerve belőlük Grapeliena bármire képes lehet. Éppen ezért úgy döntött, hogy a környék kertjeit elátkozza, és mindegyikbe hatalmas szőlő boss-t hoz létre, aki védeni fogja a kertet, hogy senki be ne tehesse a lábát.

Grapeliena tervét a környék parasztbácsijai megpróbálják szabotálni. De vajon sikerül nekik?

UML diagram

Az alábbi ábrán látható az elkészítendő osztályok adattagjai és metódusai!



Elátkozott szőlő (4 pont)

Készítsük el a `Szolo` osztályt a UML diagram alapján, amely egy elátkozott szőlőt fog reprezentálni.

- A konstruktor a nevet és erősséget várja és az adattagokat ez alapján állítja be. Kezdetben minden szőlő él.
- A *meghal* metódus a szőlő erősségét 0-ra állítja, és innentől kezdve a szőlő már nem él.
- A getterek visszaadják a nekik megfelelő adattag értékét
- Egy elátkozott szőlő erőssége sosem csökkenhet, tehát ha az erősség settere kisebb értéket kap, mint a jelenlegi erősség, akkor ne csináljon semmit (egyébként állítsa be az új értéket).

Hagyományos kert (8 pont)

Készítsd el a `Kert` osztályt, amely egy hagyományos kertet fog reprezentálni. Tudjuk a tulajdonosának a nevét (ha nincs tulajdonosa, akkor null), illetve a kert szélességét és hosszúságát cm-ben.

- Az osztály konstruktora a kert tulajdonosát, szélességét és hosszúságát várja. Ügyelj rá, hogy a szélesség és hosszúsága nem lehet 100 cm-nél kisebb, ha mégis ilyet kapnánk, akkor az adott értéket 100-ra állítsuk
- A *terulet* metódus visszaadja a kert területét méterben. Ügyelj a mértékegységekre!
- A *tulajdonosValtas* metódus paraméterben várja az új tulajdonost. Amennyiben az új tulajdonos neve megegyezik a jelenlegi tulajdonos nevével, akkor nem történik semmi, csak hamisat adunk vissza. Egyéb esetben a kapott új tulajdonos lesz a kert tulajdonosa. Ebben az esetben igazzal kell visszatérnünk.
- A getterek értelemszerűen visszaadják az adattagok értékeit.
- A toString metódus az alábbi szöveget adja vissza:

  "A kert merete: {szelesseg}x{hosszusag} cm, {tulaj} tulajdonosa."

  A {szelesseg} helyére a kert szélessége, a {hosszusag} helyére a kert hosszúsága kerüljön, míg a {tulaj} helyére a "van" szó, ha a kertnek van tulajdonosa vagy a "nincs" szó, ha a kertnek nincs tulajdonosa.

Készítsd el a *bovites* metódust is, amelynek hatására szeretnénk a kertet kibővíteni az egyik irányban, a paraméterben kapott cm-rel. A kibővítés abban az irányban történjen, amelyik hatására a kert területe nagyobb lesz. Példa:

Kert szélessége: 50, kert hosszúsága: 30, paraméter: 40.

Ha a kert szélességét növeljük, akkor az új terület ($90*30 = 2700cm^2$ lesz). Ha a kert hosszúságát növeljük, akkor az új terület $50 * 70 = 3500cm^2$ lesz. Mivel a 3500 nagyobb, mint a 2700, ezért a hosszúságot fogjuk növelni, tehát a metódushívás hatására a kert szélessége marad 50 cm, míg a hosszúsága 70 cm-re növekszik.

Elátkozott kert (8 pont)

Készítsd el az `ElatkozottKert` osztályt, amely a `Kert` osztályból származik, de egy gonosz boszorka elátkozta. Az elátkozott kert rendelkezik egy boss-szal, aki a kertben uralkodik (ő egy `Szolo`).

- Az osztály egyik konstruktora a kert tulajdonosát, szélességét és hosszúságát várja és ezek alapján inicializálja az objektumot. Ilyen esetben nincs a kertben boss, tehát a boss adattagot null-ra kell beállítani.
- Az osztály másik konstruktora a kert szélességét, hosszúságát és a boss-t várja és ezek alapján inicializálja az objektumot. Ilyen esetben a szőlő (boss) lesz a kert tulajdonosa, tehát a *tulajdonos* értékét állítsuk be a kapott boss nevére.
- Az elátkozott kert el van átkozva, ezért a széleit sem lehet pontosan behatárolni. A *terulet* metódus minden esetben a Double.NaN értékkel térjen vissza.
- Az elátkozott kertet ha bővíteni szeretnénk, akkor a területe csökken, tehát a *bovites* metódus hívja meg az ősosztály *bovites* metódusát, azonban paraméterben a kapott érték ellentettjét adjuk át!
- Ha nincs a kertben boss, akkor a *tulajdonosValtas* metódus működjön ugyanúgy, mint az ősosztályban. Ha a kertben van boss, akkor mindig hamisat adjunk vissza.

Végezetül készítsd el a *tamadas* metódust is, amely hatására a környék parasztbácsijai megpróbálják visszafoglalni a kertet és legyőzni a boss-t. A metódus paraméterben a parasztbácsik erősségét várja (int tömb). A metódus visszaadja, hogy a támadási kísérlet sikeres volt-e.

- Amennyiben a kapott tömb null vagy 0 elemű, vagy nincs boss, akkor a támadás meg sem történik, ilyenkor adjunk vissza hamisat.
- Egyéb esetben akkor sikeres a támadás, ha a parasztbácsik erősségének összege nagyobb a boss erősségénél.
- Sikeres támadás esetén:
  - a boss *meghal()*
  - a kertből eltűntetjük a boss-t (null-ra állítjuk)
  - a kertnek nem lesz tulajdonosa (ezt is null-ra állítjuk)
- Sikertelen támadás esetén a parasztbácsik erősségét tartalmazó tömb minden elemét 0-ra állítjuk.

A környék lakóinak sorsa rajtad múlik, ne hagyd őket cserben!

Jó munkát!