Programozás I. 1. zh

SZTE Szoftverfejlesztés Tanszék

2024. tavasz

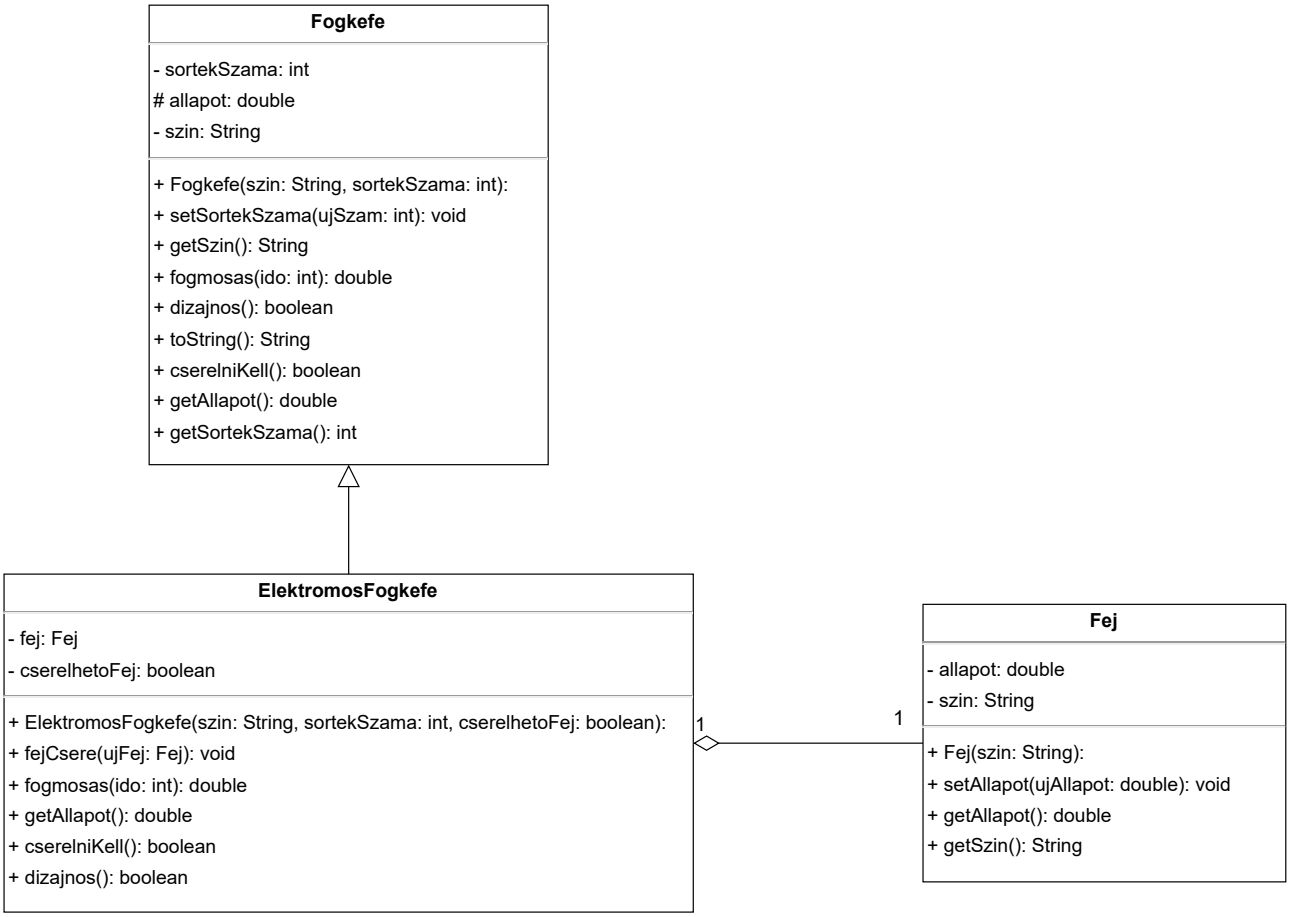
Általános követelmények, tudnivalók

- 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



Fej osztály (4 pont)

Az első osztály a `Fej`, amely egy fogkefe fejét reprezentálja. Minden fej rendelkezik egy színnel és egy állapottal, amely a használat során változhat.

A konstruktor és a szükséges getterek:

- Konstruktor, amely paraméterként megkapja a fej színét. Minden fej alapból 100%-os állapotban van.
- `getSzin()` metódus, ami visszaadja a fej színét.
- `getAllapot()` metódus, ami visszaadja a fej jelenlegi állapotát.
- `setAllapot(double allapot)` metódus, ami beállítja a fej állapotát, úgy hogy az új állapot nem lehet több az aktuális állapotnál, és nem csökkenhet 0 alá.

Fogkefe osztály (8 pont)

Készítsd el a `Fogkefe` osztályt, amely egy manuális fogkefét reprezentál. Rendelkezik színnel, sörteszámával és állapottal.

A konstruktor és a szükséges metódusok:

- Konstruktor, amely a fogkefe színét és sörteszámát várja paraméterül, ezeket rendre beállítja.
- `setSortekSzama(int sortekSzama)` metódus, amely módosítja a sörték számát, amennyiben az új érték kevesebb, mint az aktuális (fogkeféhez nem tudunk sörtét hozzáadni, azonban idővel kieshetnek belőle).
- `dizajnos()` metódus, amely megadja, hogy egy fokefe dizájnos-e. Mivel a hagoymányos fogkeféket alaposan megtervezik, ezért ez mindig igazzal térjen vissza.
- `fogmosas(int ido)` metódus, amely megadja a fogmosás hatékonyságát. A hatékonyság kiszámítása figyelembe veszi az időt (a fogmosás idejét perc), a fogkefe aktuális állapotát, és a sörteszámot. A hatékonyság csökken, ha a fogkefe állapota vagy a sörteszám csökken, és növekszik a megfelelő időtartamú használattal, egész pontosan:

  1. Az **alap hatékonyság** kiszámítása az idő alapján. Ideális időtartamnak 3 percet tekintünk. Ha az idő 3 percnél rövidebb, a hatékonyság arányosan csökken. Tehát, ha valaki például 1 percet mos fogat, akkor a hatékonyság 33.3% lesz (1/3). 3 percig tartó vagy annál hosszabb fogmosás esetén a hatékonyság 100%.
  2. A hatékonyság **módosítása a fogkefe állapotával**. A fogkefe állapota befolyásolja a tisztító hatékonyságot; egy új fogkefe (100%-os állapot) teljes hatékonysággal működik, míg egy kopottabb fogkefe (kevesebb mint 100%-os állapot) hatékonysága csökken. Az állapot hatását úgy vesszük figyelembe, hogy az alaphatékonyságot szorozzuk az állapot 50%-a plusz állapot százalékos aránya által adott szorzóval. Ez azt jelenti, hogy egy kopott fogkefe (pl. 50%-os állapot) hatékonyságát az alap hatékonyság 75%-ára módosítjuk (a számítás: $(0.5 + 0.5 * (allapot/100.0))$).
  3. A **sörteszám** közvetlenül befolyásolja a tisztító hatás erősségét. Tegyük fel, hogy az ideális sörteszám 1000, ezzel módosítjuk a hatékonyságot. Tehát a $srteszm/1000$ szeresre állítjuk be a hatékonyságot. Tehát ha a sörteszám 500, akkor a hatékonyság további 50%-kal csökken.

  Ez adja meg a hatékonyságot. Ezt követően a fogkefe **állapotának csökkentése** is feladat, mivel használjuk. Minden perc fogmosás 0.5%-kal csökkenti a fogkefe állapotát, de az állapot soha nem csökkenhet 0 alá. Ennek végén térjünk vissza a hatékonysággal.
- `cserelniKell()` metódus, amely megadja, hogy szükséges-e cserélni a fogkefét. Egy fogkefét akkor szükséges cserélni, ha az állapota 20% alá csökken.
- `toString()` metódus, amely visszaad egy szöveges reprezentációt a fogkeféről: "Egy <szin> szinu fogkefe, allapota: <allapot>", ahol a <szin> helyére a szín, az <allapot> helyére a fogkefe állapota kerül.
- Getterek az osztály adattagjaihoz: `getSzin()`, `getSortekSzama()`, és `getAllapot()` metódusok, amelyek lekérdezik a fogkefe színét, sörteszámát, és állapotát.

ElektromosFogkefe osztály (8 pont)

Készítsd el az `ElektromosFogkefe` osztályt, amely a `Fogkefe` osztályból származik. Rendelkezik fejjel, és tudjuk, hogy a fej cserélhető-e.

A konstruktor és a szükséges metódusok:

- Konstruktor, amely megkapja a fogkefe színét, sörteszámát és azt, hogy cserélhető-e a feje.
- `fejCsere(Fej f)` metódus lehetővé teszi a fogkefe fejének cseréjét, ha az cserélhető. Először is ellenőrizd, hogy a fogkefe feje cserélhető-e, és ha nem, írd ki az alapértelmezett hibacsatornára a `"Ez nem cserelheto feju fogkefe!"` hibaüzenetet. Ezt követően ellenőrizd, hogy a cserélni kívánt fej nem ugyanaz-e, mint a jelenleg használt fej. Amennyiben a jelenlegi fejet szeretnénk rátenni újra, írd ki a `"Ugyanazt a fejet nem lehet rarakni ujra!"` üzenetet az alapértelmezett hibacsatornára. Ha minden rendben van, cseréld le a fogkefe fejét a paraméterben érkezőre.
- Írd felül a `fogmosas(int ido)` metódust, amely csak az időtartamot veszi figyelembe a hatékonyság megállapításákor, hiszen az elektromos fogkefe nagyon hatékony. 1 perc fogmosás esetén 50%, 2 perc esetén 75%, míg 3, vagy több perc fogmosás esetén 100% hatékonysággal működik. A metódus csökkentse a fej állapotát a fogmosás ideje * 0.5 %-kal, valamint a fogkefe állapotát is 0.1%-kal.
- A `cserelniKell()` adjon vissza igazat, ha az elektromos fogkefe feje 10%-os állapot alatt van.
- `dizajnos()` metódus adjon vissza igazat, ha a fogkefe és a fej színe megegyezik.
- Az `getAllapot()` adja vissza a fogkefe és a fej állapotának átlagát.

Jó munkát!