Programozás I. Gyakorló feladatsor

SZTE Szoftverfejlesztés Tanszék

2023. 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 ont.
- 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.
- Az órán tanult 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")

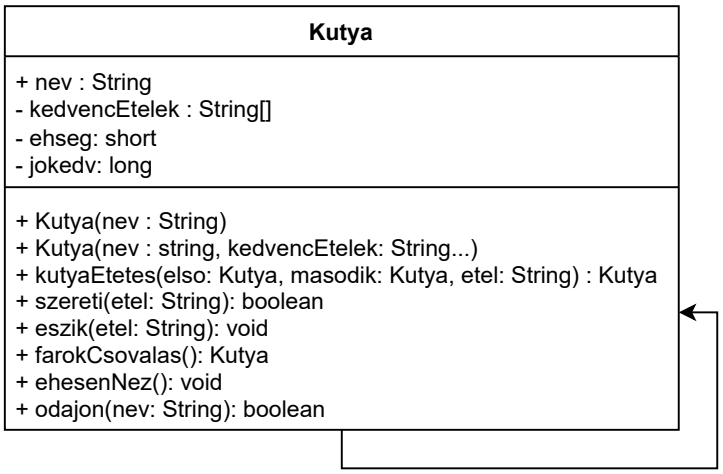
- A leírásokban bemutat példákban a stringek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!

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

Kutya osztály

Írj egy `Kutya` nevű osztályt, ami kutyákat reprezentál. Az adattagokat, valamint a szükséges metódusokat az 1. ábrán láthatjuk. Ügyelj a megfelelő láthatóságok használatára.

1. ábra. Kutya osztálydiagram



Konstruktorok

- Egy paraméteres konstruktor: a konstruktor csak a nevet várja, a kutya kedvenc ételeit inicializálja egy három elemű tömbre, amelyben a "csirke", "sajt", és "lazac" értékek vannak, hiszen ez a kutyák kedvence.
- A másik konstruktor várja a kutya nevét, valamit változó hosszúságú argumentumként a kutya kedvenc ételeit.

Metódusok

A `szereti` metódus visszaadja, hogy a kutya kedvenc ételei között van-e a paraméterben érkező étel. Figyelj rá, hogy a metódus azonos módon kezelje a "lazac" és "LaZaC" paramétereket (ha egy kutya szereti a lazacot, akkor a paraméterben érkező "LaZaC"-ot is szeretnie kell)!

Az `ehesenNez` metódus működése: növelje az éhséget, valamint írja ki, hogy "Vau!".

A `farrokCsovalas` metódus működése: növelje a jókedvet, majd térjen vissza az aktuális kutyával.

Az `eszik` metódus működése: ha a kutya nem szereti a paraméterben érkező ételt, akkor nézzen éhesen egyet, és térjen vissza a függvény. Minden más esetben csökkenjen a kutya éhsége, de legfeljebb nullára. Amennyiben az éhség csökkenése után 0 lett a kutya éhsége, hívjuk meg a `farokCsovalas` függvényt.

Az `odajon` metódus működése: a metódus térjen vissza igazzal, amennyiben a kutya odamegy a hívásra. Egy kutya akkor megy oda egy hívásra, ha a nevén szólítják, vagy legfeljebb 2 karakter különbség van a neve és a hívott név között. Ha rövidebb, esetleg hosszabb néven

hívják a kutyát, akkor arra semmi esetre se figyeljen. *Például:* ha a kutyánk neve "Jancsi", akkor a "Kancsi", "Mancsy", "Kanzsi", "Panzsi" nevekre is hallgat, és odamegy. De nem megy oda a "Sonka", "Foltoska", "Gyere ide" hívásokra.

A statikus `kutyaEtetes` metódus működése: azzal a kutyával térünk vissza, amelyik megeszi a paraméterben megadott ételt. Ha csak az egyik kutya szereti a megadott ételt, akkor ő fogja megenni, vele térjünk vissza. Ha egyik kutya sem szereti a megadott ételt, akkor térjünk vissza null értékkel. Ha mindkét kutya szereti az ételt, akkor számoljuk ki, hogy melyikük KE-faktora (KutyaÉhség) a nagyobb.

$$KE_{kutya_i} = \sqrt{\frac{ehseg_{kutya_i}}{2} * \frac{jokedv_{kutya_i}}{3}}$$

Amennyiben mindkét kutyának egyenlő a kutyaéhség faktora, akkor a paraméterben érkező első kutyával térjen vissza metódust.

Segítség a változó számú argument

A változó számú argumentumok szintaxisa `Típus...`, amelyek csak utolsó paraméterek lehetnek a függvények végén, és gyakorlatilag `Object` tömbként tudjuk használni a függvényen belül.

```
public class VarargsExample {

//    private static void pelda(int[] argumentumok) {
    private static void pelda(int... argumentumok) {
        System.out.println(argumentumok);
        int sum = 0;
        for (int i = 0; i < argumentumok.length; i++) {
            System.out.println(argumentumok[i]);
            sum += argumentumok[i];
        }
        System.out.println("SUM: " + sum);
    }

    public static void main(String[] args) {
        pelda(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    }

}
```

Jó munkát!