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.

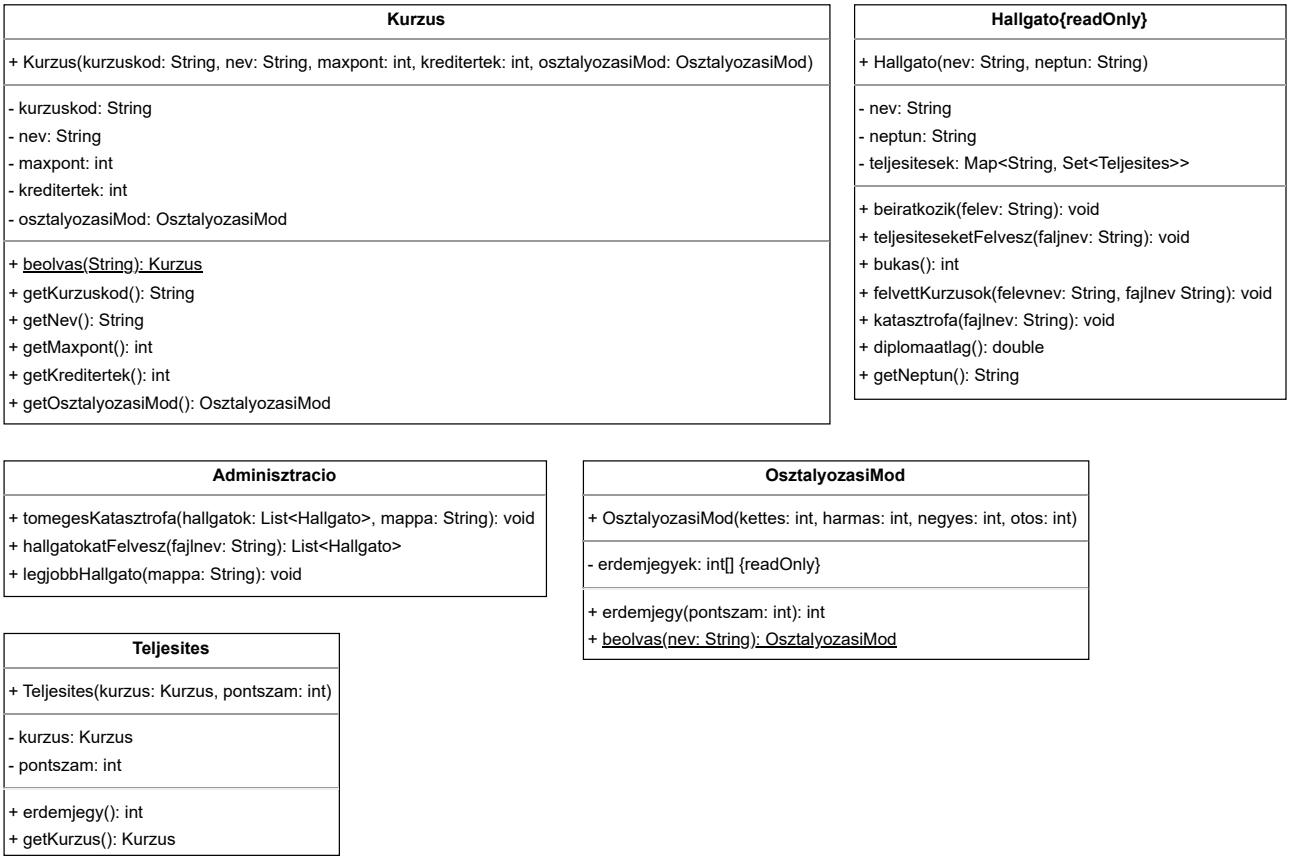
- Szövegek összehasonlításánál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")

- 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).

Hallgatói eredmények

1. ábra. Hallgatói eredmények osztálydiagram



OsztalyozasiMod osztály (5 + 1 pont)

Írj egy `OsztalyozasiMod` osztályt! Egy osztályozási mód azt mondja meg, hogy egy kurzusból adott érdemjegyhez hány pontot kell elérni. Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (1 pont)

Az érdemjegyeket tároló tömböt, egy új 4 hosszúságú megfelelő típusú tömbbel inicializáljuk. Majd ezután töltsd fel a paraméterben érkező értékekkel. A tömb első eleme a 2-eshez, a második eleme a 3-ashoz, harmadik eleme a 4-eshez, negyedik eleme az 5-öshöz szükséges pontszámot tartalmazza.

Metódusok (4 pont)

Az `erdemjegy()` metódus adja vissza, hogy az adott pontszámra az osztályzási mód által leírtak szerint hányas érdemjegy adható (1-5). **(1 pont)**

A `beolvas()` metódus olvassa be a fájlból a teljesítéshez szükséges adatokat. A fájl pontosan 4 sort tartalmaz, minden sorban 1-1 egész számmal. Az első sor tartalmazza a ketteshez, a második sor a hármashoz, a harmadik sor a négyeshez, míg a negyedik sor az ötöshöz szükséges pontszámok mennyiségét. Ezek alapján hozz létre egy `OsztalyozasiMod` objektumot, és térj vissza vele. **(3 pont)**

Kurzus osztály (6 pont)

Írj egy `Kurzus` osztályt, amely egy egyetemi kurzust reprezentál. Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (1 pont)

A kurzus adattagjait a paraméterneveknek megfelelően állítsuk be.

Getterek (1 pont)

A getter metódusok értelemszerűen a megfelelő adattagok értékeit adják vissza.

Metódusok (4 pont)

A `beolvas()` metódus olvassa be a fájlból a szükséges adatokat. A fájl két sort tartalmaz. Az első sor az alábbi formában írja le a kurzust:

kurzuskod;nev;maxpont;kreditertek

A második sora pedig egy fájlnevet tartalmaz, amelyikből az osztályozási módot kell beolvasni (célszerű felhaználni a már korábban erre megírt metódust).

Ez alapján hozz létre egy `Kurzus` objektumot, majd térj vissza ezzel az objektummal. **(4 pont)**

Teljesites osztály (3 pont)

Írj egy `Teljesites` osztályt, amely egy kurzuson nyújtott teljesítést reprezentálja. Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (1 pont)

A teljesites adattagjait a paraméterneveknek megfelelően állítsuk be.

Getterek (1 pont)

A getter metódusok értelemszerűen a megfelelő adattagok értékeit adják vissza.

Metódusok (1 pont)

Az `erdemjegy()` metódus adja vissza, hogy a hallgató ezzel az eredménnyel hányas osztályzatot szerzett. **(1 pont)**

Hallgató osztály (18 + 1 pont)

Írj egy `Hallgato` osztályt, amely egy egyetemi hallgatót reprezentál. Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságokra.

Konstruktor (2 pont)

A hallgató adattagjait a paraméterneveknek megfelelően állítsuk be, a *teljesitesek* adattagot pedig egy üres HashMap-pel inicializáljuk. Ügyelj rá, hogy a neptun kód minden esetben csak nagybetűkből és számokból állhat, továbbá hossza pontosan 6. Ha nem ilyet kapunk, akkor dobj `IllegalArgumentException` típusú kivételt a *"hibas neptun kod"* szöveggel.

Metódusok (16 pont)

A `beiratkozik()` metódus iratkoztassa be a hallgatót az adott félévre, azaz ha még nincs a teljesítések között ilyen félév, akkor hozza létre, egy üres halmazzal. Amennyiben már tartalmazza, akkor ne módosítsd a teljesítéseket, csak írd ki a standard hibacsatornára, hogy "nem lehet tobbszor beiratkozni". **(1 pont)**

A `teljesiteseketFelvesz()` metódus beolvassa a félév során teljesített kurzusokat egy fájlból (egy lehetséges fájlnév, pl. 2019_2020_2.txt). A fájl minden sora 1-1 teljesítést tartalmaz az alábbi formában:

kurzusfajl;pontszam

a *kurzusfajl* egy fájlnév, amiből a kurzus adatai beolvashatóak (a korábban létrehozott metódusok segítségével), a pontszám pedig a hallgató által elért pontszám. Ezekből hozz létre egy `Teljesites` objektumot, majd add hozzá a hallgatónak a fájlnévben megadott féléves (példa esetén 2019_2020_2) teljesítéseihez. **(4 pont)**

A `bukas()` metódus adja vissza, hogy a hallgató a tanulmányai során hány alkalommal szerzett elégtelen (1-es) értékelést. **(1 pont)**

A `felvettKurzusok()` metódus hozzon létre a paraméterben kapott néven egy fájlt, majd írja bele azoknak a kurzusoknak a nevét, amelyet a hallgató felvett az adott félévben. A kurzusneveket egymás után, külön-külön sorokba írd ki. **(4 pont)**

A `katasztrofa()` metódus hozzon létre a paraméterben kapott néven egy fájlt. A fájlba 1 sor kerüljön, mégpedig annak a félévnek a neve, amelyikben a hallgató a legtöbb kurzuson szerzett elégtelen (1-es) értékelést. Amennyiben több ilyen félév is van, akkor a *"tobb ilyen felev is van"* szöveg íródjon a fájlba. Amennyiben a hallgató egy kurzuson sem bukott meg, akkor a *"mindent teljesitett"* szöveg íródjon a fájlba. **(4 pont)**

A `diplomaatlag()` metódus adja vissza az összes felvett kurzusának a súlyozott átlagát. Súlyozott átlagot úgy tudunk számítani, hogy a kurzus érdemjegyét beszorozzuk a kreditértékével, ezeket a szorzatokat összeadjuk, majd leosztjuk az összes kredit számával. **(2 pont)**

Adminisztráció osztály (16 pont)

Metódusok (16 pont)

A `tomegesKatasztrofa()` metódus hozzon létre egy mappát a megadott névvel, majd a mappán belülre hozzon létre a listában szereplő minden hallgatóhoz egy-egy fájlt. A fájl neve a hallgató neptun kódja legyen (.student kiterjesztéssel). A fájl a hallgatónak a *katasztrofa* metódusa által legyen létrehozva, illetve az alapján legyen feltöltve. **(5 pont)**

A `hallgatokatFelvesz()` metódusnak a megadott fájlból kell olvasnia. A fájlban több sor van. Az egy sorban lévő adatok kötőjellel vannak egymástól elválasztva. Egy-egy sor az alábbiak szerint nézhet ki:

- Ha a sor a "pontszam" szóval kezdődik, akkor a (kötőjel utáni) következő adat tartalmaz egy egész számot, ami egy minimum pont lesz.
- Ha a sor a "hallgato" szóval kezdődik, akkor a (kötőjel utáni) következő három adat tartalmazza a hallgató nevét, a hallgató neptun kódját és a hallgató pontszámát. Amennyiben a hallgató pontszáma legalább akkora, mint a korábbi sorokból beolvasott minimum pontszám, akkor hozz létre egy hallgató objektumot, majd tedd be a listába.

Példa fájl:

pontszam-5
hallgato-Kis Arpad-KISAR1-5
hallgato-Nagy Andras-NA4GAA-4
pontszam-7
hallgato-Kovacs Bela-KO44AB-6
hallgato-Nagy Gyorgy-NAGYGY-7

A feldolgozás az alábbiak szerint zajlik:

- beállítjuk a minimum pontszámot 5-re
- Kis Árpádot felvesszük a listába
- Nagy Andrást nem vesszük fel a listába (nem érte el az 5-ös minimumpontot)
- a minimum pontszámot frissítjük 7-re (a korábban betett hallgatókat ez nem érinti, csak az ezutániakat)
- Kovács Bélát nem vesszük fel a listába (nem érte el a 7-es minimumpontot)
- Nagy Györgyöt felvesszük a listába **(6 pont)**

A `legjobbHallgato()` metódus a paraméterben kapott mappát dolgozza fel. A mappa több fájlt is tartalmaz. A fájlok közül azokat kell feldolgozni, amelyek .student kiterjesztésűek. Az ilyen fájlok nevei a hallgató neptun kódjával egyezik meg. A fájlban egyetlen valós szám található: a hallgató átlaga. A metódus írja ki a kapott mappán belülre, a *legjobbhallgato.txt* fájlba a legjobb hallgatónak a neptun kódját és az átlagát egy pontosvesszővel elválasztva, egy sorban.

Sajnálatos módon az adminisztrációért felelős ügyintéző nem túlságosan ért a technológiához, így néha nem közvetlenül az adott mappába menti a hallgatók fájljait, hanem létrehoz véletlenszerű almappákat (akár az almappákon belül is, és így tovább), és oda menti. Az ilyen hallgatók fájljait is vegyük figyelembe, tehát ne csak a kapott mappa gyökerében lévőket, hanem az azon belüli almappákban lévőket is. **(5 pont)**

Jó munkát!