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.

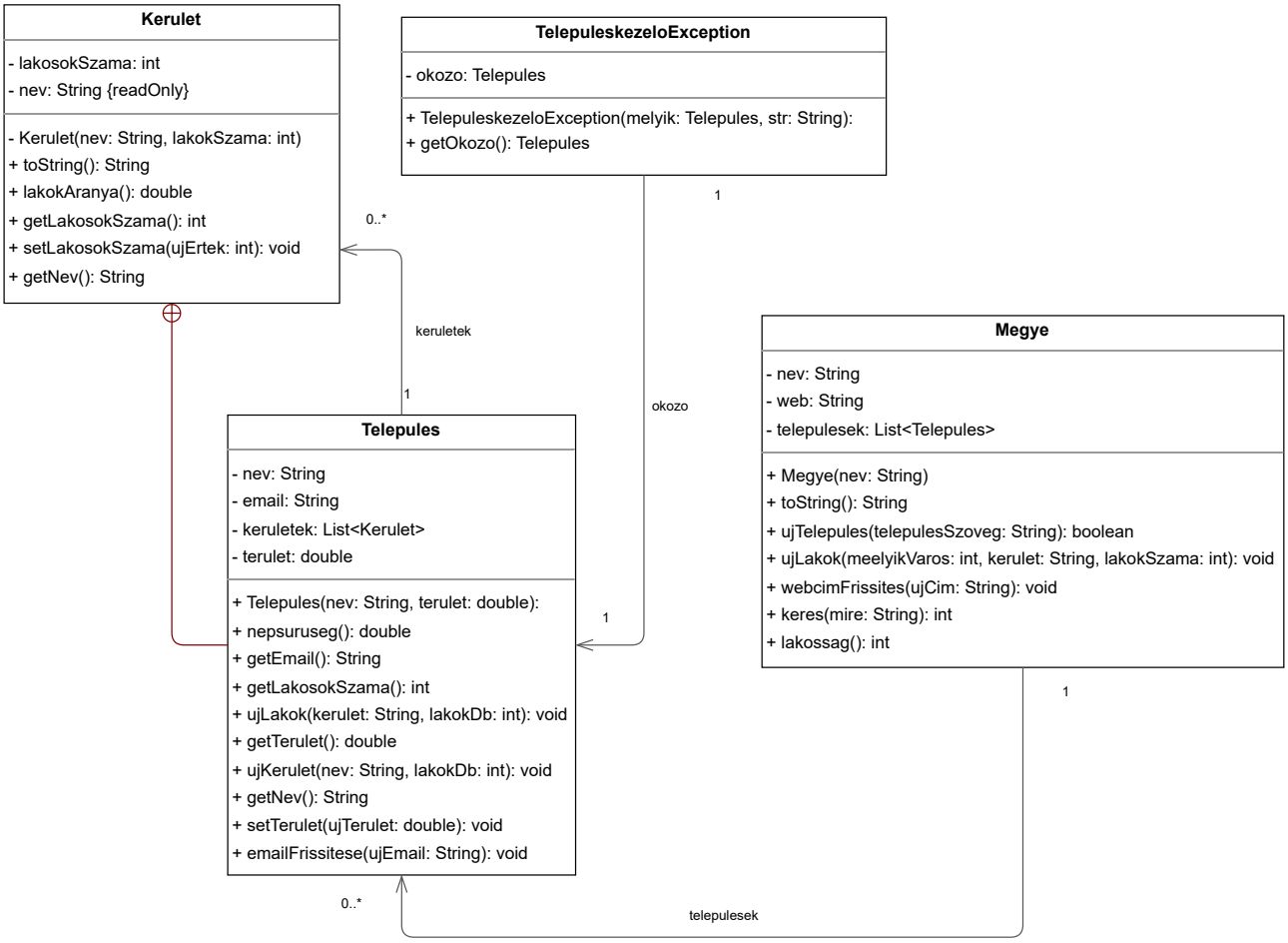
- Szövegek összehasonlításánál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")

- 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).

Településkezelő

1. ábra. Osztálydiagram



1. TelepuleskezeloException (4 pont)

Készítsd el a `TelepuleskezeloException` nevű kivételosztályt!

A paraméteres konstruktor az okozót, valamint a kivétel szövegét várja, a kapott szöveggel inicializálja ősét, az **okozo** adattagot pedig a paraméter alapján inicializálja.

2. Település (27 pont)

Készítsd el a `Telepules` nevű osztályt!

Adattagjai:

Adattag neve	Leírása
nev	a település neve
terulet	a település területe négyzetkilométerben
keruletek	a település kerületeinek listája
email	a település hivatalos e-mail címe

Ügyelj rá, hogy a *terulet* adattagot csak pozitív értékekre lehessen beállítani. Ha nem pozitív érték érkezik paraméterként, akkor állítsuk 1-re.

A konstruktor ellenőrizze le a paraméterben érkező nevet is: amennyiben nem nagy kezdőbetűvel kezdődik a név, dobjunk futásidejű, **IllegalArgumentException** típusú (beépített Java típus) kivételt, az alábbi szöveggel: "Hibas varosnev: <nev>", a <nev> helyére a beállítani kívánt (helytelen) név kerüljön. A konstruktor inicializálja a **keruletek** adattagot egy üres, tömbbel megvalósított listára. Az inicializálás során ügyelj rá, hogy a terület itt is csak pozitív lehessen, ha nem ilyen érkezik a paraméterben, állítsd 1-re. Az e-mail kezdetben null legyen.

Az `emailFrissitese` nevű metódus adott esetben *TelepuleskezeloException* típusú kivételt dobhat. A metódus ellenőrizze a paraméterben megadott e-mail címet, és figyeljen az alábbiakra:

- A beállítani kívánt e-mail címnek az "info" szöveggel kell, hogy kezdődjön.
- A beállítani kívánt e-mail cím csak ".hu"-ra végződhet.
- A beállítani kívánt e-mail címben csak egy darab "@" karakter szerepelhet.

Amennyiben ezek valamelyike nem igaz, a metódus dobjon *TelepuleskezeloException* típusú kivételt. A kivételt inicializáld az aktuális objektummal, második paraméterben pedig "Hibas e-mail cim: <email>" szöveggel. Az <email> helyett a beállítani kívánt cím legyen.
Amennyiben minden megfelelő, módosítsuk az e-mail címet!

Az `ujKerulet` nevű metódus hozzon létre egy Kerulet (lásd alább) objektumot, és adja hozzá a *keruletek* listához.

Az `ujLakok` nevű metódus adott esetben *TelepuleskezeloException* típusú kivételt dobhat.
A metódus járja be a kerületek listát, és keresse meg, hogy a paraméterben megadott nevű kerület létezik-e (figyelj rá oda, hogy az összehasonlítás ne legyen érzékeny a kis- és nagybetűkre, jelenleg a "MoRaVarOS" és a "moravaros" megegyezik). Amennyiben létezik az adott nevű kerület, növeljük meg a lakosok számát a paraméterben megadottal. Amennyiben nem létezik a kerület, dobj *TelepuleskezeloException* típusú kivételt, az alábbi szöveggel: "Nem talalhato a megadott kerulet: <kerulet>", ahol a <kerulet> helyére a paraméterben ékező kerület kerüljön.

A `getLakosokSzama` járja be a kerületeket, és adja össze a bennük lakókat, és ezzel az értékkel térjen vissza.
A `nepsuruseg` metódus a lakosok számának és a területnek a hányadosával tér vissza.

Írd meg a publikus `Kerulet` belső osztályt!

Adattag neve	Leírása
nev	a kerület neve
lakosokSzama	a kerületben lakó emberek száma

A *nev* adattagot csak egyszer, a konstruktorban lehessen beállítani, ezt garantáld!

A Kerulet `lakokAranya` metódusa térjen vissza a kerület lakosainak és a tartalmazó település összlakosságának a hányadosával.

A `toString` metódus a "<kerület neve> (<település neve>)" szöveggel térjen vissza.

4. Megye (20 pont)

Készítsd el a `Megye` osztályt.

Adattagjai:

Adattag neve	**Leírása**
nev	a megye neve
telepulesek	a megyében lévő települések listája
web	a megye weboldala

Az osztálynak egy konstruktora a A `nev` adattagot állítsa be értelemszerűen, a `telepulesek` adattagot pedig üres, tömbbel megvalósított listával inicializálja. A web adattag kezdetben legyen üres szöveg.

Az `ujTelepules` metódus feladata, hogy a szövegben szereplő adatokból egy új települést készítsen, a létrehozott településnek frissítse az e-mail címét, és ha ez sikerült, szúrja be a `telepulesek` listába. A bemenő paraméter a "nev:terulet:email" formájú (például "Szeged:281.0:info@szeged.hu"). Amennyiben nem sikerül az e-mail frissítése, térjen vissza hamis értékkel. Az esetleges kivételeket kezeld le a metóduson belül, az `ujTelepules` nem dobhat semmilyen kivételt!

A `webcimFrissites` metódus ellenőrizze, hogy a beállítani kívánt webcím tartalmazza-e a megye nevét. A kis- és nagybetűk itt sem számítanak, így ezt is vedd figyelembe az ellenőrzéskor. Amennyiben a paraméterként érkező nem tartalmazza a megye nevét, dobj *IllegalArgumentException* kivételt, az alábbi szöveggel: "Hibas webcim: <beállítani kívánt cím>". Ha a cím rendben van, módosítsd az aktuális webcímet a paraméterből érkezőre.

Az `ujLakok` metódus várja, hogy hanyadik indexű település melyik kerületébe hány új lakó érkezik. A metódus először ellenőrizze, hogy az első paraméterben érkező index létezik-e (tehát van-e egyáltalán annyi település a megyében). Ha az adott index nem létezik, dobj *IllegalArgumentException* kivételt, az alábbi szöveggel: "Nem letezik a megadott indexu varos!"
Ezt követően kérjük le a települést, és az adott településnek hívjuk meg az `ujLakok` metódusát, aminek adjuk át a kerületet, és az új lakók számát. Ez a metódus kivételt dobhat, de ezt kezeljük le. Amennyiben kivétel keletkezik, kapjuk el, majd pedig csomagoljuk be egy *IllegalArgumentException* kivételbe az alábbi módon: az *IllegalArgumentException* konstruktorának első paramétere egy szöveget vár, ez az alábbi legyen: "<okozó neve> varosban nem letezik a megadott kerulet!", ahol az okozó neve a kivétel objektumban tárolt okozó település neve legyen. A második paraméterben pedig adjuk át az elkapott kivételt.

A `keres` nevű metódus nézze meg, hogy hány olyan település van, aminek a neve tartalmazza a paraméterből érkező szöveget.

A `lakossag` metódus számítsa ki a megye településein lakók számát.

A `toString` metódus a "<név> megye (<web>)" szöveggel térjen vissza.

Jó munkát!