Programozás I. Nagy zh

SZTE Szoftverfejlesztés Tanszék

2024. tavasz - kiugró ZH

Általános követelmények, tudnivalók

- A feladat elkészítésére 90 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. feladat (6 pont)

Készítsd el az `Utas` nevű adatosztályt! Az osztályból ne lehessen további osztályokat származtatni! **(1 pont)**

Egy utasról az alábbi dolgokat tudjuk:

- az útirányát, ami megmondja, hogy melyik irányba szeretne utazni (szöveg), ennek legyen a neve *utirany*
- az úticélját, ami megmondja, hogy pontosan hová szeretne utazni (szöveg), ennek legyen a neve *uticel*
- a kedvezmény mértékét, ami megmondja mekkora kedvezményt vehet igénybe (egész szám, melynek lehetséges értékei 0, 33, 50 és 90), ennek legyen a neve *kedvezmeny*

Az osztály rendelkezzen egy *paraméteres konstruktor*ral, amely a fenti sorrendben várja a paramétereket és beállítja azokat az adattagoknak! Ellenőrizd le, hogy a kedvezmény a megadott négy értéket veszi-e fel! Ha igen, állítsd be azt, amennyiben nem állítsd 0-ra! **(3 pont)**

Az adattagok értékeit mindenki számára elérhető metódusok segítségével lehessen lekérdezni és módosítani! **(1 pont)**

A kedvezmény értékét beállító metódusnál szintén ellenőrizd, hogy a beállítani kívánt érték a megadottak közé tartozik-e. Amennyiben nem, dobj egy *IllegalArgumentException*-t, melyben a következő üzenet szerepel: "Nem megfelelo kedvezmeny: {kedvezmeny}", ahol {kedvezmeny} helyére a paraméterben kapott hibás érték kerüljön behelyettesítésre! **(1 pont)**

Megjegyzés: a többi osztály teszteléséhez elengedhetetlen ez az osztály.

2. feladat (3 pont)

Készítsd el a TulSokUtasException kivételosztályt!

A kivételnek legyen egy *helyhiany* adattagja (egész). Készítsd el a hozzá tartozó publikus lekérdezőmetódust! **(1 pont)**

A kivételnek egyetlen konstruktora legyen, amely egy egész számot vár paraméterül, amit beállít a *helyhiany* adattag értékének. A kivétel üzenetét minden esetben állítsuk be a "*Nincs hely!*" üzenetre! **(2 pont)**

Megjegyzés: a többi osztály teszteléséhez elengedhetetlen ez az osztály.

3. feladat (21 pont)

Készítsd el a `Jarmu` osztályt. Az osztályból ne lehessen objektumot létrehozni! **(2 pont)**

Egy járműről az az alábbi dolgokat tudjuk:

- a célállomását, amely megmondja, hogy milyen irányban közlekedik az adott jármű (szöveg), ennek legyen a neve *celallomas*
- az indulás idejét, amely megmondja, hogy mikor kell a járműnek elindulnia (nemnegatív egész szám), ennek legyen a neve *indulas*
- a késését, ami megmondja, hogy hány percet késik a jármű a tervezetthez képest (nemnegatív egész szám), ennek legyen a neve *keses*
- az utasait, amely megmondja, hogy milyen utasok használják a járműt (`Utas` objektumokat tartalmazó tömb), ennek legyen a neve *utasok*

Az adattagok csak az osztályon belül, illetve a gyerekosztályokból legyenek elérhetőek.

Készíts a *celallomas* és az *indulas* adattaghoz publikus metódusokat, amelyek segítségével az értékei lekérdezhetőek és módosíthatóak! A *keses* adattag értékét csak lekérdezni lehessen metódussal, módosítani ne! A többi adattaghoz ne legyen ilyen metódus! **(2 pont)**

Készíts egy *paraméteres konstruktor*t, amely a következő sorrendben vár paramétereket: *celallomas (szöveg), indulasiOra (egész), indulasiPerc (egész), kapacitas (egész)*! Ezek alapján a következőképpen inicializáld az adattagokat:

- A *celallomas*nak állítsd be a paraméterül kapott szöveget
- Az *indulas*t a következőképpen számold ki: $indulas = indulasiOra * 60 + indulasiPerc$
- A *keses* kezdőértéke legyen 0.
- Az *utasok*at egy *kapacitas* méretű üres tömbbel inicializáld. **(3 pont)**

Készítsd el a *korai* metódust, ami nem vár paramétert és egy logikai értékkel tér vissza! A metódus adjon vissza igazat, ha a jármű 7 óra előtt indul (azaz az indulás értéke 420-nál szigorúan kisebb), egyéb esetben hamisat! **(1 pont)**

Készítsd el a *szabadHelyek* metódust, ami nem vár paramétert és egy egész értékkel tér vissza! A metódus számolja meg, hogy hány szabad hely van még a járművön, azaz hány *null* érték van az *utasok* tömbben és térjen vissza ezzel a számmal. Gondoskodj róla, hogy ezt a metódust ne lehessen felülírni a gyerekosztályokban! **(5 pont)**

Készítsd el az *utasokatFelvesz* metódust, aminek paramétere egy *utas*okat tároló tömb és nem ad vissza semmit! A metódus ne legyen az osztályban megvalósítva, de jelezzük, hogy *TulSokUtasException*t dobhat! **(2 pont)**

Készítsd el az *utasokatLetesz* metódust, aminek paramétere szöveg és nem ad vissza semmit! A metódus ne legyen az osztályban megvalósítva! **(1 pont)**

Készítsd el a statikus *legpontatlanabb* metódust, ami egy *Jarmu* objektumokból álló tömböt vár paraméterben, és visszaadja közülük azt, amelyiknek a legnagyobb a késése! Amennyiben a paraméterben érkező tömb null, vagy a tömb mérete 0, akkor a visszatérési érték legyen null! Ha a tömbben minden jármű késése 0, akkor is null-t adjunk vissza! Ha több maximális késésű jármű is van a tömbben, akkor a legkisebb indexen lévőt adjuk vissza! **(5 pont)**

4. feladat (19 pont)

Készítsd el a `Busz` osztályt, ami egy speciális jármű! **(2 pont)**

Egy buszról a meglévő adatok mellett tároljuk el, hogy gyorsjárat-e (logikai típus), ennek legyen a neve *gyorsitott*. Ezt az adattagot csak az osztályon belül lehessen elérni!

Készíts a *gyorsitott* adattaghoz mindenki számára elérhető metódusokat, amelyek segítségével az értéke lekérdezhető és módosítható! **(1 pont)**

Készíts egy *paraméteres konstruktor*t, amely a következő sorrendben vár paramétereket: *celallomas* (szöveg), *indulasiOra* (egész), *indulasiPerc* (egész), *gyorsitott* (logikai). Az ősosztály konstruktora segítségével inicializálja az örökölt tagokat! Az *utasok* inicializálásához szükséges *kapacitas* minden busz esetén fixen 8! A *gyorsitott* értékének állítsd be a paraméterül kapott logikai értéket! **(2 pont)**

Definiáld felül az *utasokatFelvesz* metódust!

- A paraméterben kapott utasokat adja hozzá a busz utasait tároló tömbhöz, mindegyiket az első szabad helyre betéve, amely tömbelem még nincs utashoz rendelve!
- Az utasok sosem készítik elő a kedvezményre jogosító igazolványokat! Minden felszálló utas után, akinek a kedvezménye nem 0, növeld a késést 1-gyel!
- Amennyiben több utast kellene hozzáadni a tömbhöz, mint amennyi hely van abban, akkor adjuk hozzá az első n darab utast, aki még befér a buszba, majd dobjunk egy *TulSokUtasException*t, melyben a *helyhianyt* inicializáljuk a lemaradó utasok számával. **(8 pont)**

Definiáld felül az *utasokatLetesz* metódust! A paraméterben kapott szöveg az aktuális megálló. Szállítsuk le azokat az utasokat, akiknek ez az úticélja! Ezt tegyük meg úgy, hogy ezeket az utasokat null-ra állítjuk a tömbben. **(4 pont)**

Definiáld felül a *toString* metódust, hogy az alábbi formájú szöveget adjon vissza:

"*Ennek a {gyorsitott} busznak a celallomasa {celallomas}, {indulasOraPerc}-kor indul(t), a pillanatnyi keses {keses} perc.*"

Az *{gyorsitott}* helyére kerüljön az, hogy "gyorsitott " (szóközzel a szó után), ha a busz gyors járat, különben ne kerüljön semmi a helyére. A *{celallomas}* helyére a célállomás, a {keses} helyére a késés, míg az *{indulasOraPerc}* helyére az indulás óra-perc formátumban (Az óra az indulás 60-nal osztott egész része, a perc pedig a maradék. A vezető nullákkal nem kell foglalkoznod, tehát a 7:2 helyes formátum.) **(2 pont)**

5. feladat (26 pont)

Készítsd el a `Vonat` osztályt, ami egy speciális jármű. **(1 pont)**

Egy vonatról a meglévő adatok mellett az alábbiakat is tároljuk el:

- Inter City-e (logikai típus), ennek legyen a neve *interCity*
- elsőosztályú kocsival közlekedik-e (logikai típus), ennek legyen a neve *elsoOsztaly*

Az adattagokat csak az osztályon belül lehessen elérni.

Készíts az *interCity* adattaghoz mindenki számára elérhető metódusokat, amelyek segítségével az értéke lekérdezhető és módosítható! Az *elsoOsztaly* értékét csak lekérdezni lehessen metódus segítségével, tehát módosító metódus ne legyen! **(1 pont)**

Készíts egy *paraméteres konstruktor*t, amely a következő sorrendben vár paramétereket: *celallomas* (szöveg), *indulasiOra* (egész), *indulasiPerc* (egész), *elsoOsztalyuKocsikSzama* (egész), *masodOsztalyuKocsikSzama* (egész), *interCity* (logikai). Ezek alapján a következőképpen inicializáld az adattagokat:

- Az ősosztály konstruktora segítségével inicializálja az örökölt tagokat.
- Az *utasok* inicializálásához szükséges *kapacitas*t, azaz a tömb méretét a következőképpen kapod: $kapacitas = 5 * masodosztalyuKocsikSzama + 3 * elsoOsztalyuKocsikSzama$.
- Az *interCity* értékének állítsd be a paraméterül kapott logikai értéket.
- Az *elsoOsztaly* értéke hamis, ha az *elsoOszalyuKocsikSzama* paraméter 0, különben igaz. **(3 pont)**

Készítsd el a *kocsitHozzaad* metódust, amely két egész számot vár paraméterül (*elsoOsztalyuKocsikSzama* és *masodOszatlyuKocsikSzama*), és nem tér vissza semmivel! A paraméterek alapján növelje meg a vonat kapacitását, úgy hogy egy új tömböt foglal az utasoknak, melynek mérete $eddigiKapacitas + 5 * masodOszatlyuKocsikSzama + 3 * elsoOsztalyuKocsikSzama$! Figyelj oda, hogy az eddigi utasok átkerüljenek az új tömbbe! Ha elsőosztályú kocsit is adtak hozzá, akkor állítsuk az *elsoOsztaly* adattagot igazra! **(5 pont)**

Definiáld felül az *utasokatFelvesz* metódust!

- A paraméterben kapott utasokat add hozzá a vonat utasait tároló tömbhöz!
- Az utasok lassan szállnak fel, ezért minden 3. utas után növeld meg a késést 2-vel! (Tehát 2 utas esetén nem nő a késés, 3 utas esetén 2-vel, 5 utas esetén is 2-vel, 6 utas esetén már 4-gyel és így tovább.)
- Amennyiben több utast kellene hozzáadni a tömbhöz, mint amennyi hely van abban, akkor adjuk hozzá az első n darab utast, aki még befér a vonatba, majd dobjunk egy *TulSokUtasException*t, melyben a *helyhiany*t inicializáljuk a lemaradó utasok számával!
- Ha nincs elég hely (és Exception-t dobtunk), akkor fixen 15-tel növeld a késést (az utasok számától függetlenül)! **(8 pont)**

Definiáld felül az *utasokatLetesz* metódust! A paraméterben kapott szöveg az aktuális megálló. Szállítsuk le azokat az utasokat, akiknek ez az úticélja. Ezt tegyük meg úgy, hogy ezeket az utasokat null-ra állítjuk a tömbben. Az utasok lassan szállnak le, ezért minden 3. leszálló utas után növeld meg a késést 1-gyel! (Tehát 2 utas esetén nem nő a késés, 3 utas esetén 1-gyel, 5 utas esetén is 1-gyel, 6 utas esetén már 2-vel és így tovább.) **(5 pont)**

Készítsd el a *felsovezetekSzakadas* metódust, amely paraméterül egy egész számot vár és nem ad vissza semmit. Amennyiben a paraméter hárommal osztva 2-t ad maradékul növeld meg a késést 60-nal! Ez jelképezi a véletlen forgalmi akadályt. **(1 pont)**

Definiáld felül a *toString* metódust, hogy az alábbi formájú szöveget adjon vissza:

"*Ennek {aVonatTipusnak} a celallomasa {celallomas}, {indulasOraPerc}-kor indul(t), a pillanatnyi keses {keses} perc.*"

Az *{aVonatTipusnak}* helyére kerüljön az, hogy "az Inter Citynek", ha a vonat IC, különben a "a vonatnak" szavakat helyettesítsd be. A *{celallomas}* helyére a célállomás, míg az *{indulasOraPerc}* helyére az indulás óra-perc formátumban (Az óra az indulás 60-al osztott egész része, a perc pedig a maradék. A vezető nullákkal nem kell foglalkoznod, tehát a 7:2 helyes formátum.) **(2 pont)**

6. feladat (25 pont)

Készítsd el a `Palyaudvar` osztályt!
Egy pályaudvarról az alábbi dolgokat ismerjük:

- a tőle különböző távolságra levő településeket. Ezeknél minden település esetén a megfelelő módon eltároljuk azok távolságát ((szöveg -> egész) kulcs-érték párokat tartalmazó leképezés), a neve legyen *tavolsagok*
- az adott pályaudvarról induló járműveket (Jarmu objektumokat tároló lista), a neve legyen *jaratok*

Az adattagok legyenek mindenki számára elérhetőek.

Készíts egy *paraméter nélküli konstruktor*t, amely *tavolsagok*at egy üres piros-fekete fával megvalósított leképezéssel, a *jaratok*at pedig egy üres tömbbel megvalósított listával inicializálja. **(3 pont)**

Készítsd el a statikus *utasokatSzelektal* metódust, amely egy utasokat tartalmazó listát és egy szöveget vár paraméterül és nem tér vissza semmivel. A metódus feladata, hogy a paraméterül kapott útiránynak megfelelő utasokat a listában hagyja, minden más utast pedig kitöröljön. Tehát a kapott szöveg az útirány, egy utas pedig a listában marad, ha abba az irányba akar utazni. (Az útirány nem feltéten egyezik meg az úticéllal.) **(3 pont)**

Készítsd el a *jegyetVasarol* metódust, amely paraméterben egy `Utas` objektumot vár, és egy valós számot ad vissza. A jegy árát több tényező befolyásolja: jármű típusa, távolság, pótjegyek és a kedvezmény. Az utas minden esetben az útirányának megfelelő járműre vesz jegyet. Az garantált, hogy egy útirányba pontosan egy jármű közlekedik.

1. Az utas útiránya és a járatok célállomása alapján határozd meg a jármű típusát.
2. Az utas úticélja és a távolságok leképezés alapján határozd meg a távolságot
3. Ha vonattal utazik, akkor az árat a következő képlet alapján kapjuk:
   $ar = 20 * tavolsag * kocsiosztaly * ((100 - kedvezmeny)/100) + InterCityPotjegy$,
   ahol a kocsiosztály 1.0, ha nincs elsőosztályú kocsi, különben 1.5, az IterCityPotjegy pedig 500, ha a vonat IC, különben 0.
4. Ha busszal utazik, akkor a képlet a következő:
   $ar = 18 * tavolsag * ((100 - kedvezmeny)/100) + gyorsasagiPotjegy$,
   ahol a gyorsasagiPotjegy 175, ha a busz gyorsított, különben 0.

A metódus térjen vissza a kiszámított árral! Ügyelj a lebegőpontos számításokra a kocsiosztálynál és a kedvezménynél. **(9 pont)**

Megjegyzés: Ha csak a vonatos vagy csak a buszos képlet alapján számolsz, akkor is kaphatsz részpontszámot.

Készítsd el a statikus *utasokatBeolvas* metódust, amely paraméterben egy szöveget vár, ami egy fájlnak a neve. A metódusnak a paraméterben kapott nevű fájlból kell olvasnia. A fájl minden sora 1-1 utast ír le. Az adattagok a következő sorrendben vannak pontosvesszővel elválasztva egymástól: *utirany, uticel, kedvezmeny.* A metódus hozzon létre minden sor után egy `Utas` objektumot a megfelelő adattagokkal, majd tegye be őket egy listába, végül térjen vissza ezzel a listával. A megvalósítás során feltehetjük, hogy a fájl létezik, abban az adatok a megadott formátumban, helyesen szerepelnek. **(3 pont)**

Példa:
Budapest;Kecskemét;0
Szeged;Kiskunfélegyháza;50
Hódmezővásárhely;Hódmezővásárhely;90

Készítsd el az *utasokatBeszallit* metódust, amely egy leképezést vár paraméterül, ahol a kulcsok szövegek (útirány), az értékek pedig utasokat tartalmazó tömbök. A metódusnak nincs visszatérési értéke. A metódus feladata, hogy minden járműre (*jaratok* lista) felszállítsa az azonos útirányba utazó utasokat (azonos tömbben, minden utas azonos útirányba utazik, és a tömbben nincsenek null elemek). Ehhez használjuk a járművek *utasokatFelvesz* metódusát. Végezzük el a hibakezelést is. Ha az utasokatFelvesz metódus kivételt dobna, akkor

- ha a jármű vonat, akkor a megfelelő mennyiségű kocsit hozzáadjuk a vonathoz, de nem többet! Egy kocsiban 5 utas fér el. Ehhez használd a *kocsitHozzaad* metódust. Ilyen esetben csak másodosztályú kocsit tudnak hozzáadni a szerelvényhez.
- ha a jármű busz, akkor írjuk ki a standard hibakimenetre, hogy hány utas nem fért fel a buszra. A kiíratás után ne legyen sortörés! **(7 pont)**

Jó munkát!