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")

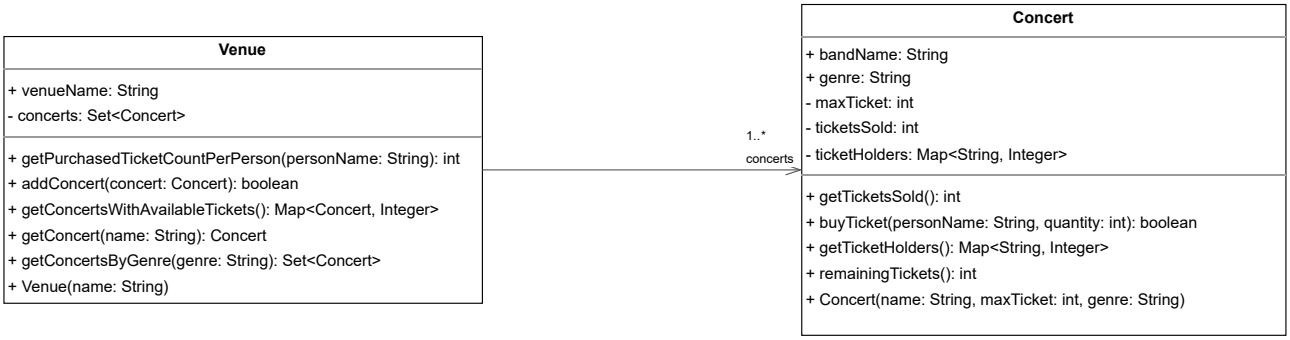
- A leírásokban bemutat példákban a stringek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!

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

Jegykezelő rendszer

A feladat egy helyszíneket és ehhez tartozó koncerteket reprezentáló rendszer megvalósítása.

1. ábra. UML osztálydiagramok



1. Concert (12 pont)

Készítsd el a `Concert` osztályt. Ez egy adott koncertet reprezentál, amire limitált számban lehet jegyeket vásárolni. A banda nevén és a stílusán kívül minden adattag privát az osztályban, az összes függvénye pedig publikus.

Adattagjai:

Adattag neve	**Leírása**
bandName	az előadó banda neve
genre	a banda stílusa
ticketsSold	az eladott jegyek száma
maxTicket	a maximálisan eladható jegyek száma
ticketHolders	a jegyeket birtoklók neve, és hány jegyük van

Konstruktorok

Az osztálynak paraméteres konstruktora legyen, amely a név, maximum vásárolható jegyeket és stílust kapja paraméterül. Ügyelj arra, hogy a konstruktorban nem paraméterül kapott értékeket is inicializáld. Például az eladott jegyek számát 0-ra. **(1 pont)**

Metódusok

A `buyTicket` segítségével vásárolhatunk jegyet! Ez a metódus paraméterben várja a vásárló nevét és a jegyek számát. Abban az esetben, ha 0 vagy kevesebb jegyet szeretne venni az illető, arra ne legyen lehetősége. Továbbá, ha a jegyek elfogytak már és nem tud annyit vásárolni, akkor térjen vissza `false` értékkel. Minden más esetben tároljuk el őket a `ticketHolders` leképezésbe. Ügyelj rá, hogy egy ember többször is vehet jegyen ugyanarra a koncertre, a tárolt

jegyek száma ebben az esetben is helyes legyen. Sikeres vásárlás esetén térjünk vissza igazzal. **(5 pont)**

A `remainingTickets` metódus adja vissza az eladható jegyek számát. **(2 pont)**

Legyen megvalósítva a `getTicketsSold` metódus. **(2 pont)**
Legyen megvalósítva a `getTicketHolders` metódus. **(2 pont)**

2. Venue (21 pont)

Készítsd el a `Venue` nevű osztályt Ez az osztály fogja végezni az ott tartandó koncertek nyilvántartását.
Figyelj rá, hogy az előző `Concert` osztály megvalósítása nélkül nem fog jól működni ez a feladatrész.

Adattagjai:

Adattag neve	**Leírása**
concerts	az adott helyen lévő koncertek
venueName	a hely neve

Konstruktorok

A konstruktor egyetlen paramétert vár, ami a hely neve lesz. Itt legyen inicializálva a koncerteket tartalmazó Set is tetszőleges implementációval. **(1 pont)**

Metódusok

Írd meg az `addConcert` metódust, mely egy paramétert vár, ami nem lehet `null`. A metódus adja hozzá az *concerts* halmazhoz a paraméterben kapott `Concert` objektumot és térjen vissza igazzal! Ha a paraméter null, ne csináljon semmit, csak térjen vissza hamissal. **(2 pont)**

Írd meg a `getConcert` metódust, amely a banda nevét várja paraméterként és visszatér azzal a `Concert` objektummal a `concerts` Set-ből; ha nem található a banda, akkor null értékkel. **(3 pont)**

Írd meg a `getConcertsByGenre` metódust, amely a banda stílusát várja paraméterként. Ez lehet például `"Melodic death metal"`, majd a meglévő `concerts` halmazból keresse ki azokat a koncerteket, amik az adott stílusban adnak elő és térjen vissza azzal a halmazzal, ami ezeket a koncerteket tartalmazza. **(5 pont)**

Írd meg a `getPurchasedTicketCountPerPerson` metódust, ami a személy nevét várja. A metódus a letárolt koncertek közül keresse ki, hova és hány jegyet vásárolt az illető, majd térjen vissza ezzel az értékkel. Például: Lehet Yuyupo vett 2 jegyet In Flames-re és 2-t Jinjer-re. A koncerteken végigiterálva kikereshető, hogy kinek hány jegye van, így a metódus visszaadja, hogy Yuyupo bizony 4 jegyet vett összesen. **(5 pont)**

Írd meg a `getConcertsWithAvailableTickets` metódus. A metódus nézze végig az összes tárolt koncertet, és amelyikre még van elérhető jegy azokat adja hozzá egy új Map-hez (a kulcs a banda neve, aki koncertezik, az érték pedig az elérhető jegyek száma), amivel

visszatér. Amennyiben egy koncert esetében az összes jegy elkelt, úgy az a koncert ne kerüljön be a visszatérési Map-be. **(5 pont)**

Jó munkát!