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).

To-do lista

1. feladat (13 pont)

Készítsünk to-do listát! Írj egy `Teendo` osztályt! Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságok használatára!

A logikai addattag beállítására hozz létre egy **atvalt** metódust, ami a logikai értéket negálja (igazból hamist, hamisból igazat csinál).

Készíts egy `default konstruktort`, amely inicializálja az adattagokat (tetszőleges értékekkel), valamint egy paraméteres konstruktort, amely a paraméterek alapján inicializálja az adattagokat. A konstruktor paramétereinek sorrendje: *nev, ido, prioritas*. Természetesen a teljesítettük adattag értéke mindig hamis, így ezt ne várja a konstruktor.

Ügyelj rá, hogy a prioritás csak 1-5 közötti egész lehet, minden más esetben írj ki egy figyelmeztető szöveget az alapértelmezett hibakimenetre, és állítsd be a prioritást 5-re.

Definiáld felül a `toString()` metódust, hogy a teendő leírását adja vissza, az alábbi formában: *"Teendo neve: {nev}, ideje: {ido}, prioritasa: {prioritas}, teljesitettuk: {igen/nem}"*.

A kapcsos zárójelek helyére az adott adattag értéke kerüljön, azonban a logikai adattag értéke helyett "igen" vagy "nem" szöveg kerüljön a toString() által visszaadott szövegbe.

Készíts egy statikus `legfontosabb` metódust, amely egy teendőkből álló tömböt vár paraméterként. A metódus térjen vissza a tömbben lévő teendők közül a legfontosabbal. A legfontosabb teendőnek 1 a prioritása. Ha több ilyen teendő van, akkor azzal térjen vissza, amelyik a tömbben alacsonyabb indexen helyezkedik el. Amennyiben üres tömböt kap a metódus, térjen vissza `null` értékkel.

2. feladat (12 pont)

Készítsük el a `Megbeszeles` nevű osztályt! Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságok használatára!

A **MEGBESZELES_DARAB** adattagot kezdetben 0-ra inicializáld, majd egy objektum létrejöttekor növeld meg 1-gyel.

Készíts `paraméteres konstruktort`, amely felhasználja az ősosztály konstruktorát is, azonban minden megbeszélés prioritása legyen 1, valamint a teendő neve az, hogy "Megbeszeles". A konstruktor paramétereinek sorrendje: *ideje, kivel, hol*.

Definiáld felül a `toString()` metódust is, az alábbi formában:
"Megbeszeles, partner: {kivel}. Idopont: {ideje}. Helyszin: {hol}"
A kapcsos zárójelek helyére az adott adattag értéke kerüljön.

Oldd meg, hogy a megbeszélések prioritását ne lehessen se csökkenteni, se növelni!

3. feladat (11 pont)

Készítsük el a `Bevasarlas` nevű osztályt! Az adattagokat, valamint a szükséges metódusokat a **??**. ábrán láthatjuk. Ügyelj a megfelelő láthatóságok használatára!

Készíts `paraméteres konstruktort`, amely felhasználja az ősosztály konstruktorát is, azonban minden bevásárlás prioritása legyen 3, valamint a teendő neve az, hogy "Bevasarlas". A konstruktor paramétereinek sorrendje: *ideje, miket, maxOsszeg*.

Készíts egy `frissit` nevű metódust, ami egy szöveget vár paraméterben. Ez adjon hozzá egy új elemet a bevásárlási listához, vesszővel elválasztva (a vessző után egy szóköz is legyen), és növelje az tervezett összegkeretet 1000 Ft-tal. Amennyiben a paraméterben érkező szöveg "<torol>", úgy a vásárlási listát módosítsd egy üres szöveggé, a tervezett összeget pedig nullázd le.

Definiáld felül a `toString()` metódust is, az alábbi formában:
"Bevasarlas. Termekek: {miket}, tervezett osszeg: {maxOsszeg} Ft"

Oldd meg, hogy a bevásárlás prioritását ne lehessen se csökkenteni, se növelni!
A kapcsos zárójelek helyére az adott adattag értéke kerüljön.

Jó munkát!

1. ábra. UML osztálydiagramok