

Programozás I. Gyakorló feladatsor

SZTE Szoftverfejlesztés Tanszék

2022. 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.
- 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).

Feltöltéskezelő

A feladat egy webes alkalmazás fájlok feltöltéséért felelős egyszerűsített részletét megvalósítani. A feltöltött fájlok a `Manager` osztályhoz kerülnek, melynek dolga kiválogatni a képeket, valamint gondoskodni róla, hogy egy mappába ne kerüljön két azonos nevű fájl.

1. FileSystemEntry (8 pont)

Készítsd el a `FileSystemEntry` absztrakt osztályt.

Adattagjai:

Adattag neve	Típusa	Leírása
parent	Folder referencia	tartalmazó mappa
name	szöveg	a bejegyzés neve

Az osztálynak egy olyan konstruktora legyen, mely ezeket az adattagokat várja paraméterként ugyanebben a sorrendben. Az adattagok beállítása után amennyiben a kapott `Folder` objektum nem *null*, hívja meg az `addChild` metódusát saját magával (lásd lentebb).

Készíts publikus lekérdező és beállító metódusokat a `name` adattaghoz.

Deklaráld az absztrakt `size` metódust, mely 64 bites egészszel tér vissza és nem vár paramétert.

Az osztálynak legyen egy `fullPath` metódusa, mely az elem teljes elérési útjával tér vissza szöveggént, paramétert nem vár. A visszaadott string a tartalmazó mappa teljes elérési útja, egy `/` jel és az adott elem neve legyen összefűzve, ha nincs tartalmazó mappa, akkor egyszerűen az elem neve.

2. Folder (5 pont)

A `Folder` osztály a `FileSystemEntry` osztály specializációja.

Saját adattagjai:

Adattag neve	Típusa	Leírása
children	<code>FileSystemEntry</code> tömb	tartalmazott fájlok és mappák

Konstruktora legyen az ősével azonos fejlécű, és annak meghívásán kívül mást ne csináljon. A `children` adattag legyen alaphól *null*.

Írd meg az `addChild` metódust, mely nem tér vissza semmivel és egy `FileSystemEntry` paramétert vár. A metódus ellenőrizze, hogy a `children` tömb *null*-e:

- ha igen, hozzon létre egy új tömböt egyetlen elemmel, ami a paraméterként kapott referencia legyen, ezt állítsa be az adattagnak.
- ha nem, hozzon létre egy új eggyel nagyobb méretű tömböt, másolja át bele a jelenlegi elemeket, majd utolsóként szúrja be a paraméterként kapottat. Ez után állítsa be az új tömböt adattagnak.

Valósítsd meg a `size` metódust. Ha a mappa üres, térjen vissza nullával, egyébként a tartalmazott elemek méretének összegével.

3. File (3 pont)

A `File` osztály a `FileSystemEntry` osztály specializációja.

Saját adattagjai:

Adattag neve	Típusa	Leírása
<code>size</code>	64 bites egész	a fájl mérete bájtban

Legyen egy háromparaméteres konstruktora, mely az őosztály paraméterein kívül a `size` adattag értékét is várja és beállítja.

A `size` metódus térjen vissza a megfelelő adattag értékével.

4. Namer (2 pont)

Hozd létre a `Namer` interfészt.

Az interfész egyetlen metódust deklarál `rename` néven, mely nem tér vissza semmivel és egy `FileSystemEntry` típusú referenciát vár paraméterként.

5. SequentialNamer (4 pont)

Az osztály implementálja a `Namer` interfészt.

Adattagjai:

Adattag neve	Típusa	Leírása
<code>index</code>	egész	számláló, hány fájlt nevezett már át az objektum

Az `index` adattag az objektum létrejöttekor legyen 1.

A `rename` metódust implementáld úgy, hogy a bejegyzés eredeti nevéhez hozzáfűzöl egy alulvonás karaktert és az `index` adattag aktuális értékét, majd ezt állítod be új névnek. Növelj egyvel az `index` értékét.

6. RandomNamer (5 pont)

Az osztály implementálja a `Namer` interfészt.

Adattagjai:

Adattag neve	Típusa	Leírása
<code>rnd</code>	<code>java.util.Random</code>	pszeudovéletlenszám-generátor
<code>length</code>	egész	milyen hosszú neveket adjon az objektum

Az adattagokon kívül legyen egy `ALPHABET` privát konstans is az osztályban az alábbi értékkel: `ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_`

A konstruktor várja a két adattagot paraméterként.

A `rename` metódust implementáld a véletlenszám-generátor és az ábécé konstans segítségével. A `java.util.Random` objektumok `nextInt(int bound)` metódusa a `[0-bound)` balról zárt, jobbról nyitott intervallumon ad vissza egy véletlen egész számot. Ennek segítségével fűzz össze az ábécéből `length` darab véletlen betűt, és azt állítsd be a paraméterként kapott fájl új nevének.

7. Manager (3 pont)

Hozd létre a `Manager` osztályt.

Adattagjai:

Adattag neve	Típusa	Leírása
<code>namer</code>	<code>Namer</code>	egyedi fájlneveket előállító objektum
<code>imagesFolder</code>	<code>Folder</code>	feltöltött képek mappája
<code>etcFolder</code>	<code>Folder</code>	feltöltött egyéb fájlok mappája

A konstruktor várja a `namer` adattagot és egy `Folder` típusú alapmappát paraméterként. Az `imagesFolder` adattagnak hozzon létre egy új `Folder` objektumot, mely a paraméterként kapott mappában van és **images** a neve, az `etcFolder`-nek ugyanígy, **etc** névvel.

Legyen egy `upload` metódusa, mely egy szöveges fájlnevet és egy 64 bites egész fájlméretet vár és `File` objektummal tér vissza. Hozzon létre egy új `File` objektumot a paraméterként kapott névvel és mérettel, a képek mappájába, ha a fájlnev `.jpg`, `.png` vagy `.gif` végződésű, az egyebek mappájába egyébként. A létrejött objektumra hívja meg a `namer` adattag `rename` metódusát. Írja ki az alapértelmezett kimenetre az alábbi három szöveget idézőjelek nélkül:

"Stored {filename} at {fullPath}"

"Images size: {imagesSize} bytes"

"Etc size: {etcSize} bytes"

Ahol a kapcsos zárójelek közti értékek a megfelelő behelyettesítések és a kiírások végén sortörés karakter szerepel. Térjen vissza a létrehozott és átnevezett `File` objektummal.

Jó munkát!